AUTOMATIC COLLECTION OF GROUP COMMUTATORS AND EXPRESSIONS

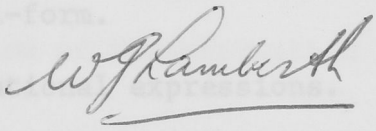
Walter J. Lamberth

Submitted to the Australian National University

for the degree of Master of Science, November 1968.

PREFACE

The work reported in this thesis is my own. Where it is the development or adaptation of the work of others acknowledgment is given.



I thank Dr M.F. Newman for his supervision. I appreciated his guidance in the difficult selection of a suitable research topic in the hybrid field of group theory and symbol manipulation and his ready availability for critical discussion.

I also thank Dr M.A. Ward and Mr J.M. Campbell for several stimulating discussions.

Computations were performed on the A.N.U. S/360 computer. The thesis was patiently typed by Mrs F. Wickland.

CONTENTS

1. INTRODUCTION.

The work reported here arose from an interest in automatic symbol manipulation in pure mathematics. The commutator calculus of group theory was chosen as the specific area of study because of its apparent suitability and its interest to local mathematicians.

One of the main problems is that of converting expressions for group elements into a canonic form. This is normally called a collecting process.

The first stage of the work consisted in producing an intuitively based set of computer procedures for executing the collecting process. This was followed by an investigation, using a recursive formalism, of the theoretical foundations for the procedures. The relationship between the formal description and the computer procedures is close. However the former sometimes avoids, in the interests of clarity, details of the computer procedures which are aimed at increased efficiency.

While gaining its inspiration and motivation from group theory this work is not on group theory but rather about the meta-problems which are accentuated by an attempt to automate the collecting process.

The theory of the commutator calculus has been given by Magnus, Karrass and Solitar [MAGNW660]† and Ward [WARDM650] but detailed knowledge

---

† The code numbers appearing in brackets refer to entries in the list of references in Section 15. The first four letters of the code are the first four letters of the leading author's last name, the fifth letter of the code is the first letter of his first name. The first two digits indicate the year of publication. The final digit is used to distinguish multiple publications in the same year. This convention is that used in the Annotated Descriptor Based Bibliography on the Use of Computers for Non-numerical Mathematics [SAMMJ661].

of this is not necessary to appreciate the resulting problem in symbol manipulation.

The relevant theorem of commutator theory is the Basis theorem of P. Hall [HALLP340]. This states:

In a free group $F(r)$ on r generators there exists an infinite sequence of commutators $C_\nu (\nu = 1,2,3,\ldots)$ of non-decreasing weights in the generators such that for $n = 1,2,3,\ldots$ and for every element W of $F(r)$,

$$W = C_1^{e_1} C_2^{e_2} \ldots C_{k(n)}^{e_{k(n)}} V_{n+1}$$

where $V_{n+1} \varepsilon F_{n+1}$ (the $(n+1)^{th}$ group of the lower central series of $F(r)$) and where the integers $e_1,\ldots,e_{k(n)}$ and V_{n+1} are uniquely determined by W. The number $k(n)$ is the number of C_ν of weight $\leq n$.

The collecting process is the algorithm for transforming the element W to its unique form.

While W is referred to as an element of a group it is more useful for purposes of manipulation to regard it as an expression which is interpretable to a group element. Then the expressions are elements of a free universal algebra in the sense of Cohn [COHNP650]. This is essentially the approach taken here but it has been found desirable in part to take the abstraction one level further and regard the expressions as strings (or equivalently words, sentences or phrases) of a formal language with the single operation of concatenation and with a set of rules which select or generate the valid strings from the set of all strings on the alphabet.

The alphabet denoted by S' of this formal language is the set $\{a,\ldots,z,0,\ldots,9,+,-,*,|,/,:,(,),\nu\}$ and the symbol ',' . With this set as generators and the operation of concatenation the semigroup S'^* is formed. The language is a certain subset of S'^*. It is defined using another subset J' of S'^* - the set of rational expressions.

The subset J' is defined formally in Section 2. Briefly and without mention of the use of parentheses they may be described as follows:

(a) $\ell,\ldots,z$ and the rational numbers are rational expressions,

(b) if λ_1 and λ_2 are rational expressions then $\lambda_1 + \lambda_2$, $\lambda_1 - \lambda_2$, $\lambda_1 * \lambda_2$ and $\lambda_1 | \lambda_2$ are rational expressions.

Of these operations the first three are the common rational operations of addition, subtraction and multiplication. The symbol '|' is the combination operator and $\lambda_1 | \lambda_2$ is the binomial coefficient and is equivalent to the more common

$$ {}^{\lambda_1}C_{\lambda_2} \quad \text{or} \quad \binom{\lambda_1}{\lambda_2} $$

which are unsuited to the restricted expression style of present day line printers. While unary negation is included in the computer procedures it is excluded from the formal description of the grammar.

The subset of rational expressions is used to define the (disjoint) subset of S'^* referred to above. This is the formal language of group expressions and is denoted by G'. It is defined formally in Section 2 and briefly as follows:

(a) $a,\ldots,k,\nu$ are group expressions,

(b) if γ_1 and γ_2 are group expressions and λ_1 is a rational

expression then $\gamma_1\gamma_2$, γ_1,γ_2 and $\gamma_1:\lambda_1$ are group expressions.

Here juxtaposition of two group expressions denotes the group product operation on the two expressions.

The operation ':' is the exponentiation operator and the expression $\gamma_1:\lambda_1$ is equivalent to $\gamma_1\gamma_1\cdots\gamma_1$ (λ_1 times) and γ_1:-1 denotes the inverse of the expression γ_1. The symbol ν is the group identity. Commutation is denoted by the operator ',' and γ_1,γ_2 is called a commutator and is defined to be $\gamma_1^{-1}\gamma_2^{-1}\gamma_1\gamma_2$. This notation departs from the conventional practice which, regardless of precedence requirements, surrounds commutators with parentheses or brackets. In this text parentheses will be used only when it is necessary to override the defined precedences.

For more complex commutators the left norming convention is used so that $(\gamma_1,\gamma_2),\gamma_3$ may be written $\gamma_1,\gamma_2,\gamma_3$.

A common convention for avoiding use of parentheses in $\gamma_1,(\gamma_2,\gamma_3)$ is to use the semicolon operator. Then $\gamma_1,(\gamma_2,\gamma_3)$ becomes $\gamma_1;\gamma_2,\gamma_3$. In this text however the semicolon is replaced by two commas. This has the advantage that it is easily generalised to more complex cases and that the additional separation of the operands assists the eye in assessing the correct structure of the expression.

For theoretical and manipulative convenience S' is augmented by the symbols '.', '$\frown$' and ',$_1$', ',$_2$',...,',$_\alpha$' where α is an arbitrarily large but finite integer. The symbol '.' becomes an

explicit group product operation, the symbol '$\frown$' an explicit concatenation operator between digits in the formation of integers and the symbol '$,_n$', for example, is a replacement symbol for a string of n juxtaposed commas. The augmented alphabet is denoted by S and the semi-group generated by S is denoted by S^*.

Associated with the augmented semi-group S^* is a set J of modified rational expressions and a set G of modified group expressions. These sets have the property that they are unambiguous context-free phrase structured languages and as such the expressions may be generated and decomposed unambiguously. This is pursued in Section 2 where the language is defined by a grammar in the computationally useful formalism called the Backus-normal-form. The grammar is useful for determining if any string of symbols on the alphabet is valid or well formed and also for providing a basis for the unique parsing or parenthesising of a grammatical expression.

The conversion of group expressions to the canonic form proceeds in two stages. In the first stage the expression is transformed to a fully parenthesised form which is then converted to the canonic form. This is described in Section 3.

The process of fully parenthesising an expression is essentially the same as the process of analysing an expression. It is a useful process in that it provides an intermediate form which is structurally and grammatically simpler and which is the basis of many proofs and definitions in the commutator calculus.

The process is described in Section 3 in a formalism due to

McCarthy [MCCAJ630]. This formalism is called a recursive conditional expression and in its own area of application, namely that of computability it follows the trend towards more expressive computer languages. It was proposed by McCarthy as a more convenient, but still theoretically well founded, algorithmic formalism than that of either Turing machines [HERMH650, DAVIM580] or Markov Normal Algorithms [MARKA620]. The means for manipulating recursive conditional expressions are available as the programming languages LISP [MCCAJ650] and CPL [FOXL660] and can be simulated to some extent in PL1.

The parenthesising process transforms the sets J and G to their fully parenthesised equivalents J^f and G^f.

A further pair of subsets of J^f and G^f are then defined in Section 4 where selection processes for determining membership of canonic subsets are given. The canonic subset of G^f is referred to by Ward as the set of basic expressions. In the case where only group products, inverses and commutators are admitted, Ward has proved that a group expression may be converted to the canonic or basic form by a finite application of the transformations. It seems reasonable to expect that a similar result would hold in the situation discussed here but this has not been proved.

Sections 5, 6 and 7 define the extended collecting process. The consequences of the extension are that the number of necessary transformations is approximately doubled and transformations for $(\gamma_1\gamma_2):\lambda$, $\gamma_1:\lambda,\gamma_2$ and $\gamma_1,\gamma_2:\lambda$ where λ is a rational expression are required. These present the difficulty that it does not appear possible to obtain the required transformation without a specific upper limit on weight. While it is believed that the inductive techniques used for

deriving these transformations could be specified algorithmically for any given upper limit on weight this has not been done. Rather, transformations which are valid for a modulus ≤ 7 have been calculated by hand and inserted into the procedures.

The PL1 procedures written to implement the collecting process bear a very close resemblance to the recursive conditional expressions and deviate significantly only in the interests of efficiency. The most significant difference is that in the computer implementation the role of the fully parenthesised expression is taken by an equivalent tree structure. This leads to rapid access to parts of an expression and very efficient modification of expressions. These procedures are discussed in Section 8 and are listed in Section 13.

It is not possible to make a generally valid statement about the effectiveness of the computer procedures. In fact the time required to collect the expression $(ab)^4$ is very much less than the time to collect the square of the collected value of $(ab)^2$. However the results leave much to be desired for while there are some computations which can be performed very much faster by machine than by hand there are other computations where the reverse is true.

To a great extent the reasons for this are understood and solutions are available and are discussed in Section 9. At the same time there is the ever present and imprecisely defined combinatorial expansion problem which will always make the heaviest demand on the ingenuity of the algorithm.

2. FORMAL GRAMMAR OF GROUP EXPRESSIONS

2.1 Introduction.

In this section the sets of rational expressions, group expressions and expressions will be defined as languages or subsets of words on an alphabet. The formal description of language is given in Section 10.4. The definitions of the various types of expressions will be given in terms of grammars which are also described formally in Section 10.4 but may be regarded informally as the information necessary either to generate all words or expressions of the language or to decide if some word on the alphabet is an element of the language.

It will be established that the group expressions form a context-free phrase structured language and that the grammar of this language is unambiguous.

2.2 The Backus-normal-form.

The formalism used to describe the alphabet and grammatical rules of the language is the Backus-normal-form otherwise known as the Backus-Naur-form or simply BNF. This notation was popularised by its use in the definition of the syntax of ALGOL-60 [NAURP600]. It gives special meaning to the following four meta-symbols:

< > ::= |

The first two are sometimes called diamond brackets and when they enclose a sequence of characters the sequence including the brackets is called a meta-variable.

The symbol '::=' separates the left and right parts of a rule. When parsing a sentence an occurrence of the left part may be replaced by the right part. The left and right parts of the rule are the components of the ordered pair described in Section 10.4.

The symbol '|' is a meta-connective and separates the right part of a rule into alternatives.

If any other symbol appears with these symbols but not enclosed in diamond brackets then that symbol stands for itself. Thus the rule

<term> ::= <factor> | <term>*<factor>

means that a <term> may be a <factor> or a <term> followed by the symbol '*' followed by a <factor>.

2.3 The grammar of rational expressions.

The expressions to be formalised consist of two disjoint parts. These consist on the one hand of expressions made up of the rational numbers and certain variables representing these numbers. On the other hand there are expressions whose elements or operands are group expression generators. The former expressions may appear only on the right hand side of the exponentiation symbol ':' while the latter may appear only on the left hand side unless surrounded by parentheses.

Corresponding to the first mentioned part there is the grammar of rational expressions.

1. <rational expression generator> ::= ℓ|m|n|o|p|q|r|s|t|
u|v|w|x|y|z

2. <digit> ::= 0|1|2|3|4|5|6|7|8|9

3. <integer> ::= <digit> | <integer><digit>

4. <rational number> ::= <integer> | <integer>/<integer>

5. <primary> ::= <rational expression generator> | <rational number>
|(<rational expression>)

6. <factor> ::= <primary> | <factor><the symbol '|'><integer>

7. <term> ::= <factor> | <term>*<factor>

8. <rational expression> ::= <term> | <rational expression>+<term>
| <rational expression>-<term>

This grammar defines the language or set of rational expressions which is denoted by the symbol 'J'.

It may be noted that because of the dual use of the symbol '|' as an operator and also as a meta-symbol it has been described in its role as an operator informally inside the meta-brackets. It will also be noted that the number and period which precedes each rule is not part of the rule but merely serves to identify it.

2.4 The grammar of group expressions.

The grammar of group expressions is now given and it can be seen that in rule 11 the grammar of rational expressions is incorporated to the right of the symbol ':'.

9. <group expression generator> ::= a|b|c|d|e|f|g|h|i|j|k|ν

10. <g-primary> ::= <group expression generator> | (<group expression>)

11. <g-factor> ::= <g-primary> | <g-factor>:<rational expression>

12. <g-term> ::= <g-factor> | <g-term><g-factor>

13. $\langle c_0 \rangle$ ::= <g-term>

14. $\langle c_\alpha \rangle$ ::= $\bigcup_{i=0}^{i=\alpha-1} \{\{\langle c_\alpha \rangle, {}_\alpha \langle c_i \rangle\} \cup \{\langle c_i \rangle, {}_\alpha \langle c_{\alpha-1} \rangle\}\}$ (see below)

15. <group expression> ::= <g-term> | $\langle c_\alpha \rangle$ for $\alpha > 0$.

This grammar denoted by G' defines the language or set of group expressions which is denoted by the symbol G'. It closely resembles the grammar actually used by the computer implementation for communication with the mathematician. In particular it may be noted that the generators are single symbols and in the definition of <g-term> the product operator is implicit.

Rule 14 of the grammar which defines the structure of commutators is unusual in that it uses the non-standard meta-symbols

$$\bigcup_{i=0}^{i=\alpha-1} \qquad \cup \qquad \{ \quad \} \qquad ,_\alpha$$

In group expressions as used by the mathematician the symbol $',_\alpha'$ is actually a string of α juxtaposed commas.

It should also be noted that in more conventional BNF notation the rule for $\langle c_\alpha \rangle$, the commutator of depth α, is really an infinite set of rules beginning:

$\langle c_1 \rangle$ ::= $\langle c_1 \rangle,\langle c_0 \rangle \mid \langle c_0 \rangle,\langle c_0 \rangle$

$\langle c_2 \rangle$ ::= $\langle c_2 \rangle,,\langle c_0 \rangle \mid \langle c_0 \rangle,,\langle c_1 \rangle \mid$
$\langle c_2 \rangle,,\langle c_1 \rangle \mid \langle c_1 \rangle,,\langle c_1 \rangle$

$\langle c_3 \rangle$ ::= $\langle c_3 \rangle,,,\langle c_0 \rangle \mid \langle c_0 \rangle,,,\langle c_2 \rangle \mid$
$\langle c_3 \rangle,,,\langle c_1 \rangle \mid \langle c_1 \rangle,,,\langle c_2 \rangle \mid$
$\langle c_3 \rangle,,,\langle c_2 \rangle \mid \langle c_2 \rangle,,,\langle c_2 \rangle$

For this reason the grammar is not strictly of the context-free type. In practice a finite limit can be placed on α which is adequate for the finite expressions encountered in any specific case.

The mathematical notation defined by the rule for $<c_\alpha>$ is, as mentioned in the introduction, also slightly unconventional. It embodies the common left norming convention which states that when precedence of operations is not defined by either the precedence of operators or the use of parentheses then the leftmost operator has precedence. So, for example a,b,c,d is to be read ((a,b),c),d. However the conventional ';' operator has been replaced by ',,' so that the expression (a,b),(c,d) is normally written a,b;c,d while here it is written a,b,,c,d. This has the advantage that it can be easily generalised to arbitrarily many commas in more complex cases and also that it assists the eye in its parsing of an expression.

It may be noted in the examples just given that while commutators are conventionally surrounded by either parentheses or brackets this is not necessary here unless the grammatically defined precedences have to be overridden. In fact the grammar specifies a precedence for all operators so that the visual obscurity associated with full parenthesisation may be avoided.

2.5 Parsing of expressions and ambiguity.

For the purposes of manipulation the expressions in $J' \cup G'$ are inconvenient because the formal processes of mathematical manipulation and the associated proofs are usually given only for a subset of $J' \cup G'$ -

the subset of fully parenthesised expressions. The process of fully parenthesising, which is the subject of the next section, is closely related to that of parsing of a sentence in a language or transforming the sentence into an explicitly structured form.

It is important to have a grammar for which there is a unique parsing process so that for any element of the language there is a unique fully parenthesised form. This unique parsing property is not achieved without care in formulation of the grammar. Suppose that the previously given definition of $\langle c_\alpha \rangle$ is replaced by the following BNF rules.

<commutator operator> ::= , | <commutator operator>,

<commutator> ::= <g-term> | <commutator>
<commutator operator><g-term> |
<commutator><commutator operator><commutator>

Using the obvious abbrevations these rules become

<co> ::= , | <co>,

<c> ::= <gt> | <c><co><gt> | <c><co><c>

This grammar will always generate valid commutators although they may possibly be supplied with a redundant number of commas. However as the example below shows it is not possible to parse or fully parenthesise a commutator unambiguously using this grammar. See Section 10.5 for a summary of the relevant theory. Ginsburg [GINSS660] gives a full discussion.

Applying this new grammar to the commutator a,,b,c yields two left-most derivations. (The use of '→' is in accord with Ginsburg's usage.)

The first derivation is as follows:

<c> → <c><co><gt>

→ <c><co><gt><co><gt>

→ <gt><co><gt><co><gt>

→ a<co><gt><co><gt>

→ a<co>,<gt><co><gt>

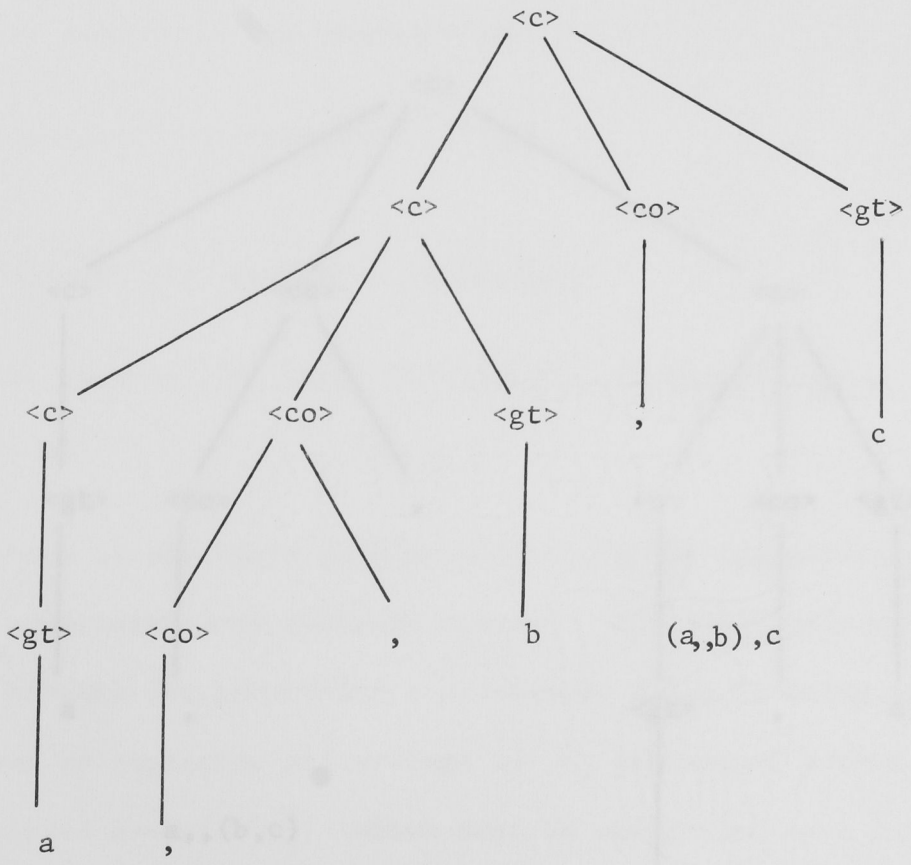
→ a,,<gt><co><gt>

→ a,,b<co><gt>

→ a,,b,<gt>

→ a,,b,c

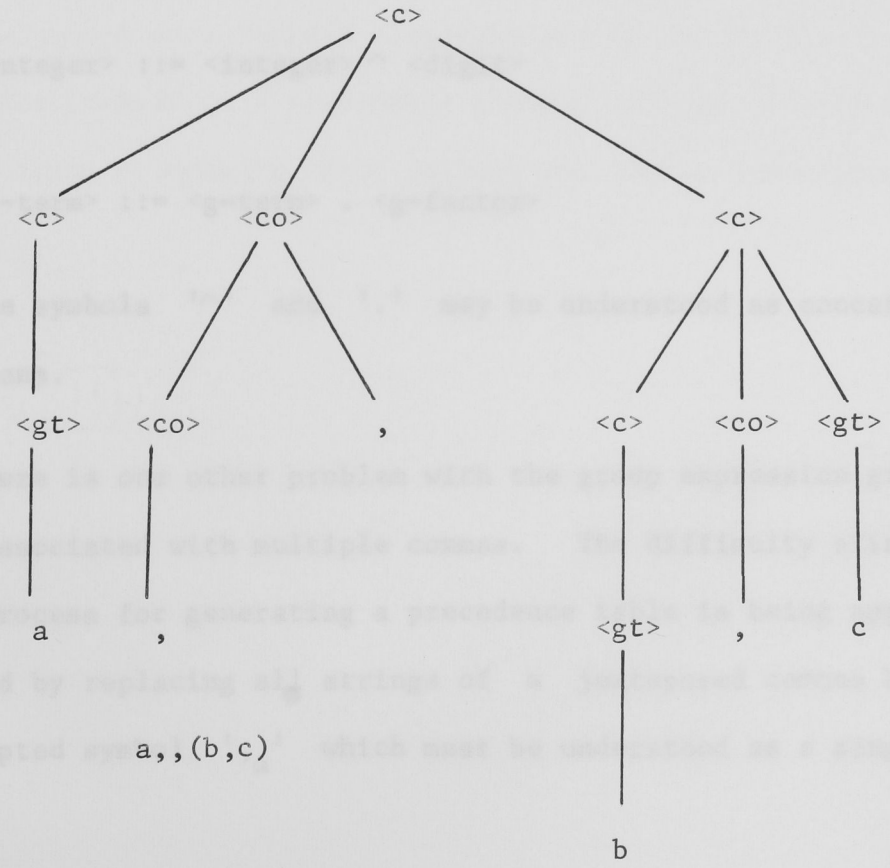
The corresponding tree structure and fully parenthesised expression are given below



The second derivation is as follows:

<c> → <c><co><c>
→ <gt><co><c>
→ a<co><c>
→ a<co>,<c>
→ a,,<c>
→ a,,<c><co><gt>
→ a,,<gt><co><gt>
→ a,,b<co><gt>
→ a,,b,<gt>
→ a,,b,c

The corresponding tree structure and fully parenthesised expression are given below.



a,,(b,c)

The ambiguity of the grammar just given is caused by the presence of juxtaposed meta-symbols in one of its rules. The existence of a unique parse structure for certain types of grammars has been proved by Floyd [FLOYR620, FLOYR621, FLOYR630] who has also established criteria for detecting ambiguity in grammars and has described algorithms for performing the parsing. A summary of these grammars is given in Section 10.

Inspection of the group expression grammar shows that there are two rules which do not conform to the restrictions of operator grammars because they contain juxtaposed meta-symbols. These are

3. <integer> ::= <integer><digit>

and

12. <g-term> ::= <g-term><g-factor>

The following simple modifications can be made to produce rules which do conform to the restrictions.

<integer> ::= <integer> ⌢ <digit>

and

<g-term> ::= <g-term> . <g-factor>

The symbols '⌢' and '.' may be understood as concatenation operations.

There is one other problem with the group expression grammar and it is associated with multiple commas. The difficulty arises when the Floyd process for generating a precedence table is being applied and is resolved by replacing all strings of α juxtaposed commas by the subscripted symbol '$,_{\alpha}$' which must be understood as a single symbol.

It is evident that a simple transformation exists for conversion from one form of group expression to the other.

Formally these processes of modification of the input/output grammar G' to the internally useful grammar which will be denoted by G consist of extending the set S' of generators of the semigroup S'^* with the symbols '.', '$\frown$' and '$,_1$', '$,_2$', ..., '$,_n$' where n is finite but arbitrarily large. The new set of generators is denoted by S and the semigroup generated by S is denoted S^*. Corresponding to this change in the grammar the set J' is transformed to a set denoted by J and the set G' is transformed to a set denoted by G.

As the set $J \cup G$ is frequently referred to in the text it will be denoted by the symbol E. The set $J' \cup G'$ is denoted E'.

With these difficulties removed it is possible to build a precedence table in accord with Floyd's algorithm and to verify that the group expression grammar is a precedence grammar and may be uniquely parsed. The resulting precedence table follows but proofs are omitted.

	(	a	:	.	'1	'2	'3	ℓ	0	⌢	/	|	*	+	-	)
)			•>	•>	•>	•>	•>					•>	•>	•>	•>	•>
a			•>	•>	•>	•>	•>									•>
:	<•		•>	•>	•>	•>	•>	<•	<•	<•	<•	<•	<•	<•	<•	•>
.	<•	<•	<•	•>	•>	•>	•>									•>
'1	<•	<•	<•	<•	•>	•>	•>									•>
'2	<•	<•	<•	<•	<•	•>	•>									•>
'3	<•	<•	<•	<•	<•	<•	•>									•>
ℓ												•>	•>	•>	•>	•>
0										•>	•>	•>	•>	•>	•>	•>
⌢									<•	•>	•>	•>	•>	•>	•>	•>
/									<•	<•		•>	•>	•>	•>	•>
|									<•	<•	<•	•>	•>	•>	•>	•>
*	<•							<•	<•	<•	<•	<•	•>	•>	•>	•>
+	<•							<•	<•	<•	<•	<•	<•	•>	•>	•>
-	<•							<•	<•	<•	<•	<•	<•	•>	•>	•>
(	<•	<•	<•	<•	<•	<•	<•	<•	<•	<•	<•	<•	<•	<•	<•	≐

Precedence table for group expression grammar.

It should be noted in connection with this table that some rows and columns are only representative of a class of symbols. In particular:

a is a representative row or column for the letters a,...,k,ν

ℓ is a representative row or column for the letters ℓ,...,z

0 is a representative row or column for the digits 0,...,9

$'_1$ $'_2$ and $'_3$ are the first three rows or columns of an arbitrarily large set.

3. THE PARENTHESISING PROCESS.

3.1 Introduction.

The group expressions defined in the previous section are in a form which is unsuited to efficient automatic manipulation. In this section functions are defined which enable the grammatically specified structure of an expression to be exposed.

The link between the grammar and the functions of this section is established by a function dlt ('dot less than'). With the use of this link three partitioning functions are defined and it is shown that the original expression may be reconstituted by appropriate concatenation of the values of the three functions.

Following this a parenthesising process is defined. It cannot be proved that this process does fully parenthesise except by reference to some other definition which must of necessity be equally precise. The given definition may, however, be checked for conformity with the intuitive notion. It is shown that the process can be performed in a finite number of steps and that it parses the expression in accord with the grammar.

The symbols μ and ϕ are used extensively in this section. The domain of μ is the set N of natural numbers and the domain of ϕ, possibly subscripted, is the set $\hat{E}$. $\hat{E}$ is the set of all expressions in E together with all words derived from expressions by deletion of one or more symbols from either end of the expression together with the set $\{\Lambda\}$. More precisely $\hat{E}$ is defined as follows:

$$\phi \in \hat{E} \iff \phi \in E \cup \{\Lambda\} \vee (\exists\psi{:}\psi \in \hat{E} \wedge (\phi = \mathrm{lr}[\psi] \vee \phi = \mathrm{rr}[\psi]))$$

where lr and rr are defined below.

The null word Λ has the role of an identity element in $\hat{E}$ with the operation of concatenation and so

$$\phi\Lambda = \Lambda\phi \quad \text{and} \quad \Lambda\Lambda = \Lambda$$

It will always be assumed that expressions submitted for processing do not contain the null word. Either it is unnecessary in an expression or the expression itself is null in which case no useful computation can be performed on it. The usefulness of the null word is internal to the functions to be defined. The convention that the leftmost or rightmost symbol of the null word is the null word will be adopted.

3.2 The partitioning functions.

Four primitive partitioning functions are defined to take the values of the single symbols at either end of a word ϕ or to take the values of the remaining words after the end symbols have been deleted. A brief mnemonic is used as the name of each function but an explanatory name is associated with the definition and appears on the right hand side of the page. As their domain and range these functions have the set $\hat{E}$.

ls[ϕ] = - left symbol

the leftmost symbol of ϕ

rs[ϕ] = - right symbol

the rightmost symbol of ϕ

lr[ϕ] = - left remainder

ϕ with the rightmost symbol deleted

rr[ϕ] = - right remainder

ϕ with the leftmost symbol deleted

Of the four functions, rs and lr are most frequently used and it is convenient to have the following shorter notation.

$$lr[\phi] \equiv \phi^{\ell}$$

$$rs[\phi] \equiv \phi^{r}$$

A length function ℓ from $\hat{E}$ to N is defined by:

$\ell[\phi]$ = - length

$$\begin{bmatrix} \phi = \Lambda \rightarrow 0; \\ \ell[\phi^{\ell}] + 1 \end{bmatrix}$$

This definition has been given in the recursive conditional expression notation of McCarthy [MCCAJ630] which is summarised in Section 10.8. The definition states that if $\phi = \Lambda$ then $\ell[\phi]$ has the value zero. Otherwise it takes the value $\ell[\phi^{\ell}] + 1$.

The level of parenthesisation of a word in $\hat{E}$ is the number of pairs of parentheses surrounding that word. In the word (a.b).(c.(c.b)).d the level of parenthesisation of d is zero while ab is at level 1 and cb at level 2. The element μ is used to indicate level of parenthesisation in the functions which follow. In these it is sometimes necessary when processing a word to ignore those parts which are enclosed in parentheses. This corresponds in Floyd's algorithm to the replacement of prime phrases by non-terminal symbols. To facilitate this type of operation the function skl is defined with domain $\hat{E} \times N$ and range $\hat{E}$. This function repreatedly replaces rightmost symbols by Λ while incrementing and decrementing μ in accord with changes in the level of parenthesisation. The process stops when μ has the value 0 and then the function takes the value of the word remaining.

Of the four functions, rs and lr are most frequently used and it is convenient to have the following shorter notation.

$$lr[\phi] \equiv \phi^{\ell}$$

$$rs[\phi] \equiv \phi^{r}$$

A length function ℓ from $\hat{E}$ to N is defined by:

$$\ell[\phi] = \qquad \text{- length}$$

$$\left[\begin{array}{l}\phi = \Lambda \rightarrow 0; \\ \ell[\phi^{\ell}] + 1\end{array}\right]$$

This definition has been given in the recursive conditional expression notation of McCarthy [MCCAJ630] which is summarised in Section 10.8. The definition states that if $\phi = \Lambda$ then $\ell[\phi]$ has the value zero. Otherwise it takes the value $\ell[\phi^{\ell}] + 1$.

The level of parenthesisation of a word in $\hat{E}$ is the number of pairs of parentheses surrounding that word. In the word (a.b).(c.(c.b)).d the level of parenthesisation of d is zero while ab is at level 1 and cb at level 2. The element μ is used to indicate level of parenthesisation in the functions which follow. In these it is sometimes necessary when processing a word to ignore those parts which are enclosed in parentheses. This corresponds in Floyd's algorithm to the replacement of prime phrases by non-terminal symbols. To facilitate this type of operation the function skl is defined with domain $\hat{E} \times N$ and range $\hat{E}$. This function repreatedly replaces rightmost symbols by Λ while incrementing and decrementing μ in accord with changes in the level of parenthesisation. The process stops when μ has the value 0 and then the function takes the value of the word remaining.

skl[ϕ;μ] = - skip level

$$\begin{bmatrix} \phi = \Lambda \rightarrow \Lambda; \\ \mu = 0 \rightarrow \phi; \\ \phi^r =) \rightarrow \text{skl}[\phi^\ell;\mu+1]; \\ \phi^r = (\rightarrow \text{skl}[\phi^\ell;\mu-1]; \\ \text{skl}[\phi^\ell;\mu] \end{bmatrix}$$

The operation of this function may be demonstrated by applying it to the element a,(b,(c,d) of $\hat{E}$. If this element were encountered during the analysis of a group expression, μ would have the value 1 corresponding to the presence of an unmatched left parenthesis.

skl[a,(b,(c,d);1]
= skl[a,(b,(c,d ;2]
= skl[a,(b,(c, ;2]
= skl[a,(b,(c ;2]
= skl[a,(b,(;2]
= skl[a,(b, ;1]
= skl[a,(b ;1]
= skl[a,(;1]
= skl[a, ;0]
= a,

The function dlt (dot less than) is now defined. This has domain $\hat{E} \times \hat{E}$ and its range is the set T of truth values. In the definition the grammatical relation '<·' is used and is to be understood as follows. If ϕ_1 and ϕ_2 ε S then $\phi_1 <\cdot \phi_2$ if and only if the entry in the precedence table in Section 2 corresponding to the row indexed by ϕ_1 and the column

indexed by ϕ_2 contains the symbol '$<\cdot$'.

It may be noted that $\Lambda <\cdot \phi_2$ for all $\phi_2 \varepsilon S$. For strings of arbitrary length dlt takes the value T if there exists a symbol in ϕ_1 at zero level of parenthesisation which $<\cdot$ some symbol of ϕ_2 at zero level of parenthesisation.

$dlt[\phi_1;\phi_2]$ = - dot less than

$$\left[\begin{array}{l} \phi_1 = \Lambda \rightarrow T;\quad \phi_2 = \Lambda \rightarrow F; \\ \phi_1 \varepsilon S \wedge \phi_2 \varepsilon S \wedge \phi_1 <\cdot \phi_2 \rightarrow T; \\ \phi_1 \notin S \wedge \phi_1^r =) \rightarrow dlt[skl[\phi_1^\ell;1];\phi_2]; \\ \phi_2 \notin S \wedge \phi_2^r =) \rightarrow dlt[\phi_1;skl[\phi_2^\ell;1]]; \\ \phi_1 \notin S \wedge (dlt[\phi_1^\ell;\phi_2] \vee dlt[\phi_1^r;\phi_2]) \rightarrow T; \\ \phi_2 \notin S \wedge (dlt[\phi_1;\phi_2^\ell] \vee dlt[\phi_1;\phi_2^r]) \rightarrow T; \\ F \end{array}\right]$$

Provided $\phi \varepsilon E$ the next three functions op', lp', and rp' define three unique subwords of the word ϕ such that

$$lp'[\phi]op'[\phi]rp'[\phi;0] = \phi.$$

While these functions are intended to partition members of E their domains are $\hat{E}$ or $\hat{E} \times N$ because in the process of evaluation they may recursively reference themselves with words which are not in E.

It should be noted that $lp'[(\phi)] = \Lambda$, $rp'[(\phi);0] = (\phi)$

and $op'[(\phi)] = \Lambda$. In other words the functions lp' and rp' do not remove redundant external parentheses.

$op'[\phi]$ = - operator'

$$\begin{bmatrix} \phi = \Lambda \to \Lambda; \quad \phi\varepsilon S \to \Lambda; \\ \phi^r =) \to op'[skl[\phi^{\ell};1]]; \\ dlt[\phi^{\ell};\phi^r] \to op'[\phi^{\ell}]; \\ \phi^r \end{bmatrix}$$

$lp'[\phi]$ = - left part'

$$\begin{bmatrix} \phi = \Lambda \to \Lambda; \quad \phi\varepsilon S \to \phi; \\ \phi^r =) \to lp'[skl[\phi^{\ell};1]]; \\ dlt[\phi^{\ell};\phi^r] \to lp'[\phi^{\ell}]; \\ \phi^{\ell} \end{bmatrix}$$

$rp'[\phi;\mu]$ = - right part'

$$\begin{bmatrix} \phi = \Lambda \to \Lambda; \quad \phi\varepsilon S \to \Lambda; \\ \phi^r =) \to rp'[\phi^{\ell};\mu+1]); \\ \phi^r = (\to \ rp'[\phi^{\ell};\mu-1](; \\ \mu>0 \vee dlt[\phi^{\ell};\phi^r] \to rp'[\phi^{\ell};\mu]\phi^r \\ \Lambda \end{bmatrix}$$

It should be emphasised again that the words to be partitioned must be members of E. In this case all parentheses will be matched and invalid function references such as rp'[(;0] cannot occur since when counting parentheses from the right there can never be an excess of opening parentheses over closing parentheses.

The uniqueness of the partitioning and its conformity to the structure defined by the grammar result from the use of the function dlt together with right to left processing.

An example of the action of rp' on the expression $a,_2 b,c.(a,b)$ is given.

rp'[$a,_2 b,c.(a,b);0$]
= rp'[$a,_2 b,c.(a,b;1]$)
= rp'[$a,_2 b,c.(a, ;1]b$)
= rp'[$a,_2 b,c.(a ;1],b$)
= rp'[$a,_2 b,c.(;1]a,b$)
= rp'[$a,_2 b,c. ;0](a,b)$
= rp'[$a,_2 b,c ;0].(a,b)$
= rp'[$a,_2 b, ;0]c.(a,b)$
= rp'[$a,_2 b ;0],c.(a,b)$
= rp'[$a,_2 ;0]b,c.(a,b)$
= $\Lambda b,c.(a,b)$
= $b,c.(a,b)$

Theorem:

$$lp'[skl[\phi;\mu]]op'[skl[\phi;\mu]]rp'[\phi;\mu] = \phi$$

for all $\phi \varepsilon \hat{E}$ and all $\mu \varepsilon N$.

Proof:

The theorem is proved by induction on the length of ϕ, that is $\ell[\phi]$. When $\ell[\phi] = 0$ the theorem is obviously true. Let n be greater than 0 and suppose that it is true for all $\psi \varepsilon \hat{E}$ such that $\ell[\psi] < n$. It is to be shown that it is true for all $\phi \varepsilon \hat{E}$ such that $\ell[\phi] = n$. The conditions stated for the various cases below are not necessarily mutually exclusive and it is assumed that case i implies not case j where $j < i$. This conforms with the conditional expression definitions.

Case 1. $\mu = 0 \wedge \phi \varepsilon S$

$$\begin{aligned} & lp'[skl[\phi;\mu]]op'[skl[\phi;\mu]]rp'[\phi;\mu] \\ = \; & lp'[\phi]op'[\phi]rp'[\phi;0] \\ = \; & \phi\Lambda\Lambda \\ = \; & \phi \end{aligned}$$

Case 2. $\mu = 0 \wedge \phi^r =)$

$$\begin{aligned} & lp'[skl[\phi;\mu]]op'[skl[\phi;\mu]]rp'[\phi;\mu] \\ = \; & lp'[skl[\phi^\ell;1]]op'[skl[\phi^\ell;1]]rp'[\phi^\ell;1]) \\ = \; & \phi^\ell) \\ = \; & \phi \end{aligned}$$

Case 3. $\mu = 0 \wedge \phi^r = ($

This cannot occur.

Case 4. $\mu = 0 \wedge dlt[\phi^\ell;\phi^r]$

$$\begin{aligned} & lp'[skl[\phi;\mu]]op'[skl[\phi;\mu]]rp'[\phi;\mu] \\ = \; & lp'[\phi]op'[\phi]rp'[\phi;0] \\ = \; & lp'[\phi^\ell]op'[\phi^\ell]rp'[\phi^\ell;0]\phi^r \\ = \; & \phi^\ell\phi^r \\ = \; & \phi \end{aligned}$$

Case 5. $\mu > 0 \wedge \phi^r =)$

$$\begin{aligned} & lp'[skl[\phi;\mu]]op'[skl[\phi;\mu]]rp'[\phi;\mu] \\ = \; & lp'[skl[\phi^\ell;\mu+1]]op'[skl[\phi^\ell;\mu+1]]rp'[\phi^\ell;\mu+1]) \\ = \; & \phi^\ell) \\ = \; & \phi \end{aligned}$$

Case 6. $\mu > 0 \wedge \phi^r = ($

$$\begin{aligned} & lp'[skl[\phi;\mu]]op'[skl[\phi;\mu]]rp'[\phi;\mu] \\ = \ & lp'[skl[\phi^{\ell};\mu-1]]op'[skl[\phi^{\ell};\mu-1]]rp'[\phi^{\ell};\mu-1](\\ = \ & \phi^{\ell}(\\ = \ & \phi \end{aligned}$$

Case 7. $\mu > 0 \wedge dlt[\phi^{\ell};\phi^r]$

$$\begin{aligned} & lp'[skl[\phi;\mu]]op'[skl[\phi;\mu]]rp'[\phi;\mu] \\ = \ & lp'[skl[\phi^{\ell};\mu]]op'[skl[\phi^{\ell};\mu]]rp'[\phi^{\ell};\mu]\phi^r \\ = \ & \phi^{\ell}\phi^r \\ = \ & \phi \end{aligned}$$

Case 8.

$$\begin{aligned} & lp'[skl[\phi;\mu]]op'[skl[\phi;\mu]]rp'[\phi;\mu] \\ = \ & lp'[skl[\phi^{\ell};\mu]]op'[skl[\phi^{\ell};\mu]]rp'[\phi^{\ell};\mu]\phi^r \\ = \ & \phi^{\ell}\phi^r \\ = \ & \phi \end{aligned}$$

Corollary:

$$lp'[\phi]op'[\phi]rp'[\phi;0] = \phi \quad \text{if} \quad \phi \in E$$

Proof:

$$\begin{aligned} & lp'[\phi]op'[\phi]rp'[\phi;0] \\ = \ & lp'[skl[\phi;0]]op'[skl[\phi;0]]rp'[\phi;0] && \text{- by definitions} \\ = \ & \phi && \text{- by the theorem} \end{aligned}$$

Note that the restriction on ϕ is necessary to ensure matched parentheses for skl and rp' and also to ensure a grammatical structure for dlt in case 4 of the theorem.

3.3 The parenthesising function.

A function is now given for fully parenthesising, in a non-redundant fashion, any expression in E. This function $\underline{f}$ has as its domain the set $\hat{E} \times \hat{E} \times N$. When $\underline{f}$ is first referenced ϕ_1 must be a member of E, $\phi_2 = \Lambda$ and $\mu = 0$. The range of $\underline{f}$ is denoted by E^f and is the union of two disjoint subsets J^f and G^f which are respectively the fully parenthesised rational and group expressions.

$f[\phi_1;\phi_2;\mu]$ = - fully parenthesise

$$\left[\begin{array}{l}
\phi_1 = \Lambda \wedge \phi_2 = \Lambda \rightarrow \Lambda; \\
\phi_1 \varepsilon S \wedge \phi_2 = \Lambda \rightarrow \phi_1; \\
\phi_1 = \Lambda \wedge \phi_2^r =) \wedge ls[\phi_2] = (\rightarrow f[rr[\phi_2^\ell];\Lambda;0]; \\
\phi_1^r =) \rightarrow f[\phi_1^\ell;)\phi_2;\mu+1]; \\
\phi_1^r = (\rightarrow f[\phi_1^\ell;(\phi_2;\mu-1]; \\
\mu>0 \rightarrow f[\phi_1^\ell;\phi_1^r\phi_2;\mu]; \\
dlt[\phi_1^\ell;\phi_1^r] \rightarrow f[\phi_1^\ell;\phi_1^r\phi_2;0]; \\
\phi_1^\ell \varepsilon S \wedge \phi_2 \varepsilon S \rightarrow \phi_1^\ell \phi_1^r \phi_2; \\
\phi_1^\ell \varepsilon S \wedge \phi_2 \notin S \rightarrow \phi_1^\ell \phi_1^r (f[\phi_2;\Lambda;0]); \\
\phi_1^\ell \notin S \wedge \phi_2 \varepsilon S \rightarrow (f[\phi_1^\ell;\Lambda;0])\phi_1^r \phi_2; \\
(f[\phi_1^\ell;\Lambda;0])\phi_1^r (f[\phi_2;\Lambda;0])
\end{array}\right]$$

Note that because of the action of lines 3,4 and 5 of the definition the fully parenthesised expressions are free of redundant parentheses. The fully parenthesised form of ((a)) is a.

As mentioned in Section 2 the multiple commas of E' are replaced in E by a subscripted comma which is interpreted as a single symbol.

In the computer procedures the process of fully parenthesising is followed by a function which deletes the subscripts from the commas.

To assist in the description of certain properties of $\underline{f}$ the words computation and terminate are defined.

A computation of a function is a sequence of steps in the evaluation of the function. Each step consists of the replacement of one function reference by the appropriate value obtained from the definition. A computation is said to terminate if there exists a step for which there is no further function to be evaluated. The function is said to terminate finitely if the number of steps in the computation is finite. The techniques and language of computability theory [DAVIM580, HERMH650] are appropriate here but have not been used as their range of application in the text is too limited to justify the extensive theoretical modifications necessary. Changes would appear desirable since the domains and ranges most frequently encountered are subsets of $\hat{E}$ rather than the set of positive integers used in conventional computability theory. McCarthy [MCCAJ630] has given an informal presentation of a suitable approach.

That a computation of the parenthesising process when applied to an expression in E does terminate in a finite number of steps may be seen by observing that if a function reference $f[\phi_1;\phi_2;\mu_1]$ is evaluated in any one step of a computation as $\psi_3 f[\psi_1;\psi_2;\mu_2]\psi_4$ where ψ_3 and ψ_4 are any elements of S^* then $2\ell[\psi_1] + \ell[\psi_2]$ is always less than $2\ell[\phi_1] + \ell[\phi_2]$. $\underline{\ell}$ is the previously defined length function.

Apart from showing that $\underline{f}$ terminates it is necessary to show

that it does parenthesise in accord with the grammar. This will be demonstrated for the operation of exponentiation. Other cases can be treated similarly.

Suppose that a <g-factor>, a ':' and a <rational expression> are concatenated as specified by rule 11 of the grammar:

<g-factor>:<rational expression>

and let this resulting expression be denoted by ϕ. It will be shown that the parenthesising function partitions ϕ about the (rightmost) colon as required by the grammar.

First note that $\underline{f}$ inspects symbols in a right to left scan skipping over parenthesised expressions and when $\mu = 0$ (i.e. when at zero level of parenthesisation) seeking out the first (and therefore rightmost) symbol which is less than (i.e. $<\cdot$) all symbols to the left of it at zero level of parenthesisation.

Now it will be seen by reference to the precedence table in Section 2 that ':' is dot less than any of the symbols $\ell,\ldots z,0,\ldots,9,+,-,*,|,\uparrow,/$ which, apart from parentheses, are the only symbols appearing in rational expressions. It will also be noted that $\underline{f}$ does scan past parentheses when determining the manner of partitioning at zero level of parenthesisation.

Finally, it will be seen that all the symbols which can appear in a <g-factor> at zero level of parenthesisation, namely $a,\ldots,k,\nu,:$, are not dot less than ':'.

Therefore the rightmost colon will be taken by f as the point for partitioning of ϕ or as the 'outermost' operator of ϕ and this is the type of structure generated by the grammatical rule:

<g-factor> ::= <g-factor> | <g-factor>:<rational expression>

The following sample computations illustrate the effect of f

(a) $f[a,_2b,c;\Lambda;0]$

$= f[a,_2b,\ ;c;0]$

$= f[a,b;,c;0]$

$= f[a,_2;b,c;0]$

$= a,_2(f[b,c;\Lambda;0])$

$= a,_2(f[b,;c;0])$

$= a,_2(b,c)$

(b) $f[a,b,_2c;\Lambda;0]$

$= f[a,b,_2;c;0]$

$= (f[a,b;\Lambda;0]),_2c$

$= (f[a,;b;0]),_2c$

$= (a,b),_2c$

4. CANONIC EXPRESSIONS.

4.1 Introduction.

In Section 3 the method for converting expressions in $\mathcal{E}$ into the fully parenthesised form $\mathcal{E}^f$ was described. In this section and those to follow the set $\mathcal{E}^f$ will be of major interest. The functions of this section serve to define a subset of $\mathcal{E}^f$ the expressions of which will be called canonic.

There are two distinguishable aspects to the definition. The first is associated with the structure or relationship of operators of expressions while the second is associated with the relationship of operands to one other. Structurally, canonic expressions take the form of products of powers of commutators. That is, the innermost or first executed operators must be commutator operators and the outermost operators must be product operators. A similar type of structure is required of the exponents themselves. The second aspect of the definition is concerned with the ordering of expressions and cannot be simply described.

The following extracts from Section 10 are relevant here.

(a) '[' and ']' are meta-brackets

(b) Greek lower case letters, possibly subscripted, are used as variables.

(c) Roman upper case script letters denote sets

(d) ';' is the meta-delimiter

(e) When referred to in text non alpha-numeric symbols are surrounded by single quotes.

The set notation {<x>} used here is intended to denote the set of all derivations of the meta-variable <x>.

Set Name	Description
N'	{<integer>} ∪ {∞}
Q	{<rational number>}
B	{<rational expression generator>}
R	$\{B \cup Q\}$
J	{<rational expression>}
J^f	fully parenthesised subset of J
L	{<group expression generator>}
G	{<group expression>}
G^f	fully parenthesised subset of G
E	$\{G \cup J\}$
E^f	fully parenthesised subset of E
O	{:;/;|;*;+;-;.;,;↑}
T	{T;F}

The domains of the variables (possibly subscripted) to be used are:

Variable	Domain
X	O
ρ	G^f
σ	J^f
τ	E^f

The following additional conventions and notations are used.

(f) '<', '>' and '=' are relations on the integers or rational members and have the normal meaning.

(g) '=' is assumed to have an obvious meaning in comparing any expressions.

(h) $'\overset{A}{<}'$ and $'\overset{A}{>}'$ are standard alphabetic orders.

4.2 Preliminary functions.

In many cases the functions to be defined are mutually recursive and for this reason it is not always possible to avoid reference to a particular function before it is defined. The definitions begin with three functions of basic importance. These are the operator function and the left and right part functions. These functions have for their domain the set of fully parenthesised expressions and they divide such expressions into the three obvious parts. They are defined in terms of the previously given functions op', lp' and rp' and differ from them in that the unprimed functions remove the pair of external parentheses if they are present. These functions are defined in the notation of recursive conditional expressions described in Section 10.8.

op[τ] = op'[τ] - operator

lp[τ] = - left part

$$\left[\begin{array}{l} ls[lp'[\tau]] = (\ \wedge\ rs[lp'[\tau]] =)\ \rightarrow\ lr[rr[lp'[\tau]]]; \\ lp'[\tau] \end{array}\right]$$

rp[τ] = - right part

$$\left[\begin{array}{l} ls[rp'[\tau;0]] = (\ \wedge\ rs[rp'[\tau;0]] =)\ \rightarrow\ lr[rr[rp'[\tau;0]]]; \\ rp'[\tau] \end{array}\right]$$

length[σ] = - length

$$\left[\begin{array}{l} \sigma\varepsilon R \rightarrow 1; \\ \underline{\text{length}}[\text{lp}[\sigma]]+\text{length}[\text{rp}[\sigma]] \end{array}\right]$$

The domain of this function is the set J^f and its range is the set N. It determines the number of operands in a rational expression. The function prd (product) has range T and takes the value T if and only if the expression σ is free of operators other than '*' or '$\frown$'. Function rat (rational) also has range T and takes the value T if and only if the argument is a rational number or has imbedded within it a rational number.

prd[σ] = - product

$$\left[\begin{array}{l} \sigma\varepsilon R \rightarrow \text{T}; \\ (\text{op}[\sigma] = * \vee \text{op}[\sigma] = \frown\ ) \wedge \text{prd}[\text{lp}[\sigma]] \wedge \text{prd}[\text{rp}[\sigma]] \rightarrow \text{T}; \\ \underline{\text{F}} \end{array}\right]$$

rat[σ] = - rational

$$\left[\begin{array}{l} \sigma\varepsilon Q \rightarrow \text{T}; \\ \underline{\text{rat}}[\text{lp}[\sigma]] \vee \text{rat}[\text{rp}[\sigma]] \end{array}\right]$$

Functions lo (left operand) and ro (right operand) have for their domain the set $E^f \times O$ and their range is the set E^f. They select respectively the leftmost operand and the rightmost operand of an expression with respect to an operator. If, for example, τ = ((a,b)∘(c,d)).((e,f).g) then lo[τ;.] = a,b and ro[τ;.] = g.

lo[τ;X] = - left operand

$$\left[\begin{array}{l} \tau\varepsilon RUL \rightarrow \tau; \\ op[\tau]\neq X \rightarrow \tau; \\ lo[lp[\tau];X] \end{array}\right]$$

ro[τ;X] = - right operand

$$\left[\begin{array}{l} \tau\varepsilon RUL \rightarrow \tau; \\ op[\tau]\neq X \rightarrow \tau; \\ ro[rp[\tau];X] \end{array}\right]$$

The next two functions to be given are complementary to lo and ro. Thus if τ has the value given in the example above then le[τ;.] = ((a,b).(c,d)).(e,f). These functions differ from all previous functions in that they produce the desired value by composition or concatenation while the previous functions have used decomposition only.

le[τ;X] = - left expression

$$\left[\begin{array}{l} op[\tau]\neq X \rightarrow \tau; \\ rp[\tau]\varepsilon RUL \vee op[rp[\tau]]\neq X \rightarrow lp[\tau]; \\ lp'[\tau]X(le[rp[\tau];X]) \end{array}\right]$$

re[τ;X] = - right expression

$$\left[\begin{array}{l} op[\tau]\neq X \rightarrow \tau; \\ lp[\tau]\varepsilon RUL \vee op[lp[\tau]]\neq X \rightarrow rp[\tau]; \\ (re[lp[\tau];X])Xrp'[\tau;0] \end{array}\right]$$

4.3 Canonic rational expressions.

With the auxiliary functions now defined the five functions

cp (canonic product), ltp (less than product), lpr (left product), rpr (right product) and ce (canonic expression) can be given. Together these constitute the definition of a canonic exponent. For simplicity only three operators are permitted in a canonic exponent. These are the sum, difference and product operators. Structurally the canonic exponent is in the form of sums of products and since exponents themselves may not have exponents the products may consist of repetitions of the same rational expression generator.

Within products ordinary alphabetic ordering is used for rational expression generators. The rational number coefficient, if any, appears at the right hand end of the product. The ordering of operands of a summation is done first by length of the operands and for operands of the same length alphabetic ordering is used. The unary minus operation is not permitted.

cp[σ] = - canonic product

$$\left[\begin{array}{l} \sigma\varepsilon\mathcal{R} \rightarrow T; \\ \sim prd[\sigma] \rightarrow F; \\ rat[le[\sigma;*]] \rightarrow F; \\ \sim[cp[lp[\sigma]] \wedge cp[rp[\sigma]]] \rightarrow F; \\ ro[lp[\sigma];*] \underline{\overset{A}{\leq}} lo[rp[\sigma];*] \rightarrow T; \\ \underline{F} \end{array}\right]$$

$ltp[\sigma_1;\sigma_2]$ = - less than product

$$\left[\begin{array}{l} \sigma_1 \varepsilon Q \wedge \sigma_2 \varepsilon Q \to F; \\ \sigma_1 \varepsilon B \wedge \sigma_2 \varepsilon R \to T; \\ ro[\sigma_1;*] \varepsilon Q \to ltp[le[\sigma_1;*];\sigma_2]; \\ ro[\sigma_2;*] \varepsilon Q \to ltp[\sigma_1;le[\sigma_2;*]]; \\ length[\sigma_1] < length[\sigma_2] \to T; \\ lo[\sigma_1;*] \overset{A}{<} lo[\sigma_2;*] \to T; \\ ltp[re[\sigma_1;*];re[\sigma_2;*]] \end{array}\right]$$

$lpr[\sigma]$ = - left product

$$\left[\begin{array}{l} \sigma \varepsilon R \to \sigma; \\ op[\sigma] = * \to \sigma; \\ op[\sigma] = + \to lpr[lo[\sigma;+]]; \\ op[\sigma] = - \to lpr[lo[\sigma;-]] \end{array}\right]$$

$rpr[\sigma]$ = - right product

$$\left[\begin{array}{l} \sigma \varepsilon R \to \sigma; \\ op[\sigma] = * \to \sigma; \\ op[\sigma] = + \to rpr[ro[\sigma;+]]; \\ op[\sigma] = - \to rpr[ro[\sigma;-]] \end{array}\right]$$

ce[σ] = - canonic exponent

```
[σεR → T;
 prd[σ] ∧ cp[σ] → T;
 ∿ce[lp[σ]] → F;
 ∿ce[rp[σ]] → F;
 ltp[rpr[lp[σ]];lpr[rp[σ]]] → T;
 F]
```

4.4 Canonic group expressions

It is now convenient to define the weight of a group expression by the function wt which maps the group expressions G^f to the set N' of integers and the symbol '∞'. It is to be understood that in all arithmetic involving the set N' the following rules apply:

(a) $i+\infty = \infty$

(b) $i-\infty = 0$

(c) $\infty \pm i = \infty$

(d) $\infty+\infty = \infty$

(e) $\infty-\infty = 0$

wt[ρ] = - weight

```
[ρ = ν → ∞;  ρεL → 1;
 op[ρ] = : → wt[lp[ρ]];
 op[ρ] = . →
    [wt[lp[ρ]]<wt[rp[ρ]] → wt[lp[ρ]];
     wt[rp[ρ]]];
 op[ρ] = , → wt[lp[ρ]]+wt[rp[ρ]]]
```

Note that in this function there is an imbedded or nested conditional expression. It is evaluated in the same way as any of the conditional expressions which in other cases have been used to define functions. Function wt acts as a coarse order on group expressions.

The binary function lt (less than) uses the function wt and two auxiliary functions to define an order on group expressions. The domain of lt is the set $G^f \times G^f$ and its range is the set T. The auxiliary functions are ld and tr.

ld[ρ] = - leading part

[lt[lp[ρ];rp[ρ]] → rp[ρ];lp[ρ]]

tr[ρ] = - trailing part

[lt[lp[ρ];rp[ρ]] → lp[ρ];rp[ρ]]

$lt[\rho_1;\rho_2]$ = - less than

$$\left[\begin{array}{l}
\rho_1 \varepsilon\ L \wedge \rho_2 \varepsilon L \wedge\ \rho_1 \overset{A}{<} \rho_2 \rightarrow T; \\
wt[\rho_1] < wt[\rho_2] \rightarrow T; \\
wt[\rho_1] > wt[\rho_2] \rightarrow F; \\
op[\rho_1] = : \rightarrow lt[lp[\rho_1];\rho_2]; \\
op[\rho_2] = : \rightarrow lt[\rho_1;lp[\rho_2]]; \\
op[\rho_1] = . \wedge lt[lp[\rho_1];\rho_2] \wedge lt[rp[\rho_1];\rho_2] \rightarrow T; \\
op[\rho_2] = . \wedge lt[\rho_1;lp[\rho_2]] \wedge lt[\rho_1;rp[\rho_2]] \rightarrow T; \\
lt[ld[\rho_1];ld[\rho_2]] \rightarrow T; \\
ld[\rho_1] = ld[\rho_2] \wedge lt[tr[\rho_1];tr[\rho_2]] \rightarrow T; \\
ld[\rho_1] = ld[\rho_2] \wedge tr[\rho_1] = tr[\rho_2] \\
\qquad \wedge lt[lp[\rho_2];lp[\rho_1]] \rightarrow T;F
\end{array}\right]$$

Function com takes the value T if the argument whose domain is the set G^f has no operator other than ','. Such an expression is a commutator.

$com[\rho]$ = - commutator

$$\left[\begin{array}{l}
\rho \varepsilon L \rightarrow T; \\
op[\rho] = , \wedge com[lp[\rho]] \wedge com[rp[\rho]] \rightarrow T; \\
F
\end{array}\right]$$

The definition of the function canonic can now be given. It has domain G^f and range T. This function takes the value T for arguments from a certain subset of G^f. Such expressions are said to be canonic. As remarked earlier the significance and uniqueness of a similar canonic form are dealt with by Ward.

canonic[ρ] = - canonic

```
⌈ ρεL → T;
| ∿canonic[lp[ρ]] → F;
| op[ρ] = : ∧ com[lp[ρ]] ∧ ce[rp[ρ]] → T;
| ∿canonic[rp[ρ]] → F;
| op[ρ] = . ∧ lt[lp[ρ];rp[ρ]] → T;
| com[ρ] →
|     ⌈ lt[rp[ρ];lp[ρ]] → T;
|     | wt[lp[ρ]]>1 ∧ [lt[rp[lp[ρ]];rp[ρ]]
|     |                ∨ rp[lp[ρ]] = rp[ρ]]
|     |       → T;
|     ⌊ F                                    ⌋;
⌊ F                                          ⌋
```

5. ROLE OF THE MODULUS.

In Sections 6 and 7 the collecting process proper will be described. It will be shown to consist of a number of transformation rules together with a description of the manner in which they are to be applied. There is one rule, however, which is so central to the process that it will be discussed separately. This rule is concerned with an integer called the modulus which is associated with any group expression that is to be collected or transformed to the canonic form. The value of the integer is supplied by the mathematician together with the group expression to be collected. The mathematical significance of this modulus is not discussed here but may be found in P. Hall [HALLP340] and Magnus, Karrass and Solitar [MAGNW660].

The computational significance of the modulus is that all commutators of weight greater than or equal to the modulus 'reduce' to the identity element or 'are equivalent' to that element. This process is analogous to modular arithmetic. It requires careful application for two reasons. The first is that timely application will effect a vast improvement in speed of calculation and the other is that delayed application may result in a non-terminating process. The Jacobi transformation (see Section 6) would be susceptible to this latter problem.

Ideally then all transformations should be regarded as applicable modulo a certain weight (called the modulus) and in fact terms which would later reduce to the identity should not be generated in the

first place. While this is the approach of the computer procedures the description here delays the reduction process in the interests of clarity.

There is a further aspect of modular transformations. To discuss this it is convenient to use the word subexpression. This is a part of an expression which is itself an expression. Now while certain subexpressions have weight less than the modulus supplied for the main expression they may eventually contribute sufficient weight to a subexpression containing them to cause the generation of expressions whose weight is not less than the modulus of the main expression. An example of this is given. A hand collection of ab,c,d modulo weight 4 might take the form:

$$
\begin{aligned}
ab,c,d &= (a,b)(a,c,b)(b,c),d \\
&= (a,b,d)(a,b,d,(a,c,b)(b,c))((a,c,b)(b,c),d) \\
&= (a,b,d)\nu((a,c,b)(b,c),d) \\
&= (a,b,d)((a,c,b)(b,c),d) \\
&= (a,b,d)(a,c,b,d)(a,c,b,d,,b,c)(b,c,d) \\
&= (a,b,d)\nu\nu(b,c,d) \\
&= (a,b,d)\nu(b,c,d) \\
&= (a,b,d)(b,c,d)
\end{aligned}
$$

A more efficient method is to recognise in the second line that the subexpression a,c,b in its context is already 'too heavy' and may be replaced by the identity even though its weight is only 3. Then the calculation is as follows:

$$
\begin{aligned}
ab,c,d &= (a,b)(a,c,b)(b,c),d \\
&= (a,b)\vee(b,c),d \\
&= (a,b)(b,c),d \\
&= (a,b,d)(a,b,d,,b,c)(b,c,d) \\
&= (a,b,d)\vee(b,c,d) \\
&= (a,b,d)(b,c,d)
\end{aligned}
$$

While this example illustrates the principle it does not demonstrate the immense practical importance of this method. In more complex expressions practical computation would not be possible unless this method were used.

In the following functions the reduction process is formalised. Function <u>modl</u> enables the calculation of a <u>submodulus</u> for the left part of an expression whose modulus is known. Function <u>modr</u> operates similarly for the right part. The domain of these functions is the set $G^f \times N'$ and their range is the set N'. The process of replacing expressions whose weight is not less than the modulus of that expression by the identity is performed by function <u>red</u>.

modl[τ;μ] = - left modulus

$$
\left[\begin{array}{l}
\overline{\text{op}[\tau] = .} \rightarrow \mu; \\
\text{op}[\tau] = : \rightarrow \mu; \\
\underline{\text{op}[\tau] = ,} \rightarrow \mu\text{-wt}[\text{rp}[\tau]]
\end{array}\right]
$$

modr[τ;μ] = - right modulus

$$
\left[\begin{array}{l}
\overline{\text{op}[\tau] = .} \rightarrow \mu; \\
\text{op}[\tau] = : \rightarrow \infty; \\
\underline{\text{op}[\tau] = ,} \rightarrow \mu\text{-wt}[\text{lp}[\tau]]
\end{array}\right]
$$

$red[\tau;\mu]$ = - reduce

$$\left[\begin{array}{l} \tau\varepsilon L \rightarrow \tau; \\ \tau\varepsilon J^{f} \rightarrow \tau; \\ wt[\tau]\geq\mu \rightarrow \nu; \\ \underline{red[lp'[\tau];\mu]op[\tau]red[rp'[\tau;0];\mu]} \end{array}\right]$$

6. TRANSFORMATION RULES.

6.1 Introduction.

In this section the transformation rules to be used for converting an expression of E^f into the canonic form are given. These rules are classified broadly on the basis of the lowest precedence operation appearing in an expression before it is transformed. So the function trp is concerned with product transformations. The classification is arbitrary from a description viewpoint but it does lead to more efficient manipulation.

The predicates appearing in the functions take the value T if the expression on the left of the equality has the form of the right hand side of the equality. For example the predicate $\rho = \rho_1{:}\text{-}1\rho_2$ has the value T if and only if there exist expressions ρ_1 and ρ_2 such that $\rho_1{:}\text{-}1\rho_2$ is the same expression as ρ. If such a predicate has the value T the function takes as its value the expression to the right of the arrow. Each such component of the function can be regarded as a transformation rule.

In many cases the function scan appears within the transformation rule. This function will be defined in the next section. It should suffice to state here that this function maps an element from the domain $E^f \times N'$ into the subset of E^f consisting of canonic expressions. It is applied within the transformation function to ensure that all subexpressions produced by the transformation are converted to canonic form.

In the transformation functions the domain of the variable ρ, possibly subscripted, is the set of fully parenthesised group expressions G^f and the domain of σ is the set of fully parenthesised rational expressions J^f.

While it is required that the arguments of the functions be fully parenthesised the transformation function is permitted to generate non fully parenthesised expressions. This has been done to maintain clarity and simplicity. The reason for specifying the restricted domain for the arguments is also that of simplicity. If non parenthesised expressions were permitted the transformations involving commutation would have a complexity similar to that of the relevant grammar. The middle course chosen is convenient for another reason - to the mathematician the non-parenthesised expression or non fully parenthesised expression is more easily interpreted while, within the computer, the fully parenthesised expression, particularly as a list tree, is more easily manipulated. Thus the expressions specifying the transformations must be fully parenthesised in a manner similar to that described in Section 3 for the input expressions themselves. In practice while the parenthesising process is applied automatically to the input expressions the equivalent operation on the transformations has been done once and for all by hand.

It is a necessary assumption in what follows that the left and right parts of the expressions to be transformed are in canonic form. Section 7 will show how this condition is ensured.

For simplicity the unary minus operation has not been included in the grammar of group expressions or in the function f. When unary minus is used in the transformations in the form '$-\sigma$' it should be understood as $0-\sigma$. Unary operations are permitted in the computer procedures.

6.2 Group expression transformation.

The group product transformation trp has domain $G^f \times N'$.

trp[ρ;μ] = - transform product

$$\left[\begin{array}{l}
\rho = \rho_1{:}{-1}\rho_1 \rightarrow \nu;\quad \rho = \rho_1\rho_1{:}{-1} \rightarrow \nu; \\
\rho = \rho_1\nu \rightarrow \rho_1;\quad \rho = \nu\rho_1 \rightarrow \rho_1; \\
\rho = \rho_1\rho_2 \rightarrow \rho_1{:}2; \\
\rho = \rho_1{:}\sigma\rho_1 \rightarrow \mathrm{scan}[\rho_1{:}(\sigma{+}1);\mu]; \\
\rho = \rho_1\rho_1{:}\sigma \rightarrow \mathrm{scan}[\rho_1{:}(\sigma{+}1);\mu]; \\
\rho = \rho_1{:}\sigma_1\rho_1{:}\sigma_2 \rightarrow \mathrm{scan}[\rho_1{:}(\sigma_1{+}\sigma_2);\mu]; \\
\rho = \rho_1\rho_2 \wedge \mathrm{lt}[\mathrm{lo}[\rho_2;.];\mathrm{ro}[\rho_1;.]] \\
\qquad \rightarrow \mathrm{scan}[\mathrm{le}[\rho_1;.]\mathrm{lo}[\rho_2;.]\mathrm{ro}[\rho_1;.] \\
\qquad\qquad (\mathrm{ro}[\rho_1;.],\mathrm{lo}[\rho_2;.])\mathrm{re}[\rho_2;.];\mu]; \\
\rho
\end{array}\right]$$

Some explanation of the eighth rule may assist understanding. If ρ_1 and ρ_2 are irreducible to the product operation and if $\mathrm{lt}[\rho_2;\rho_1]$ then trp is to take the value $\mathrm{scan}[\rho_2\rho_1(\rho_1,\rho_2);\mu]$. On the other hand if ρ_1 is reducible and ρ_2 is irreducible ρ_1 may be written as $\rho_3\rho_4$ where ρ_4 is irreducible. In this case the function

takes the value $scan[\rho_3\rho_2\rho_4(\rho_4,\rho_2);\mu]$ if $lt[\rho_2;\rho_4]$. The other two possibilities are treated in a similar manner.

The commutation transformations are grouped in the function trc which has a similar domain and range to trp.

$trc[\rho;\mu]$ = - transform commutator

$$
\begin{array}{l}
\rho = \rho_1,\rho_1 \rightarrow \nu;\quad \rho = \rho_1,\nu \rightarrow \nu;\quad \rho = \nu,\rho_1 \rightarrow \nu; \\
\rho = \rho_1{:}{-1},\rho_2 \rightarrow scan[(\rho_1,\rho_2){:}{-1}(\rho_2,\rho_1,\rho_1{:}{-1});\mu]; \\
\rho = \rho_1,\rho_2{:}{-1} \rightarrow scan[(\rho_1,\rho_2){:}{-1}(\rho_2,\rho_1,\rho_2{:}{-1});\mu]; \\
\rho = \rho_1\rho_2,\rho_3 \rightarrow scan[(\rho_1,\rho_3)(\rho_1,\rho_3,\rho_2)(\rho_2,\rho_3);\mu]; \\
\rho = \rho_1,\rho_2\rho_3 \rightarrow scan[(\rho_1,\rho_3)(\rho_1,\rho_2)(\rho_1,\rho_2,\rho_3);\mu]; \\
\rho = \rho_1,\rho_2 \wedge lt[\rho_1;\rho_2] \wedge com[\rho_1] \wedge com[\rho_2] \rightarrow (\rho_2,\rho_1){:}{-1}; \\
\rho = \rho_3,\rho_2,\rho_1 \wedge lt[\rho_1;\rho_2] \wedge lt[\rho_2;\rho_3] \wedge com[\rho_1] \\
\qquad \wedge com[\rho_2] \wedge com[\rho_3] \\
\qquad \rightarrow scan[(\rho_1,\rho_3,,\rho_3,\rho_2)(\rho_3,\rho_2,,\rho_2,\rho_1,\rho_3) \\
\qquad\quad (\rho_2,\rho_1,\rho_3){:}{-1}(\rho_3,\rho_2,,\rho_2,\rho_1)(\rho_2,\rho_1,,\rho_1,\rho_3,\rho_2) \\
\qquad\quad (\rho_1,\rho_3,(\rho_3,\rho_1,\rho_2){:}{-1})(\rho_3,\rho_1,\rho_2) \\
\qquad\quad (\rho_2,\rho_1,,\rho_1,\rho_3)(\rho_1,\rho_3,,\rho_3,\rho_2,\rho_1);\mu]; \\
\rho = \rho_1,\rho_2{:}\sigma \rightarrow scan[(\rho_2,\rho_1){:}{-}\sigma(\rho_2,\rho_1,\rho_2){:}{-}(\sigma|2) \\
\qquad\quad (\rho_2,\rho_1,\rho_2,\rho_2){:}{-}(\sigma|3)(\rho_2,\rho_1,\rho_2,\rho_2,\rho_2){:}{-}(\sigma|4) \\
\qquad\quad (\rho_2,\rho_1,\rho_2,\rho_2,\rho_2,\rho_2){:}{-}(\sigma|5) \\
\qquad\quad (\rho_2,\rho_1,\rho_2,,\rho_2,\rho_1){:}{-}((\sigma{+}1)|3) \\
\qquad\quad (\rho_2,\rho_1,\rho_2,\rho_2,,\rho_2,\rho_1){:}{-}((\sigma{+}1)|4);\mu]; \\
\rho = \rho_2{:}\sigma,\rho_1 \rightarrow scan[(\rho_2,\rho_1){:}\sigma(\rho_2,\rho_1,\rho_2){:}(\sigma|2)(\rho_2,\rho_1,\rho_2,\rho_2){:}(\sigma|3) \\
\qquad\quad (\rho_2,\rho_1,\rho_2,\rho_2,\rho_2){:}(\sigma|4)(\rho_2,\rho_1,\rho_2,\rho_2,\rho_2,\rho_2){:}(\sigma|5) \\
\qquad\quad (\rho_2,\rho_1,\rho_2,,\rho_2,\rho_1){:}(\sigma|3{+}(\sigma{+}1)|3) \\
\qquad\quad (\rho_2,\rho_1,\rho_2,\rho_2,,\rho_2,\rho_1){:}(\sigma|4{+}2{*}(\sigma{+}1)|4);\mu]; \\
\rho
\end{array}
$$

The ninth transformation here is known as the Jacobi identity. The next two transformations are valid only when $\mu \leq 7$. This is not a serious limitation as it is possible to provide a transformation which is valid up to any desired value of μ. Krause [KRAUE640] has described a similar calculation for $(\rho_1\rho_2):\sigma$. It is also possible to provide a transformation which is valid for arbitrarily large values of μ provided $\sigma \varepsilon N$.

It may be noted that in this as in other transformation functions if a match is not achieved the function takes the value of the expression supplied as an argument.

The function for transforming exponent expressions has no new features and is given without further comment.

tre[ρ;μ] = - transform exponent

```
ρ = ρ1:0 → ν;ρ = ρ1:1 → ρ1;ρ = ν:σ → ν;
ρ = ρ1:σ1:σ2 → scan[ρ1:(σ1*σ2);μ];
ρ = (ρ1ρ2):-1 → scan[ρ2:-1ρ1:-1;μ];
ρ = (ρ1,ρ2):σ → scan[ρ1:σρ2:σ(ρ2,ρ1):(σ|2)(ρ2,ρ1,ρ1):(σ|3)
                     (ρ2,ρ1,ρ2):(σ|2+2*σ|3)(ρ2,ρ1,ρ1,ρ1):(σ|4)
                     (ρ2,ρ1,ρ1,ρ2):(2*σ|3+3*σ|4)
                     (ρ2,ρ1,ρ2,ρ2):(2*σ|3+3*σ|4)
                     (ρ2,ρ1,ρ1,ρ1):(σ|5)(ρ2,ρ1,ρ1,ρ1,ρ2):(3*σ|4+4*σ|5)
                     (ρ2,ρ1,ρ1,ρ2,ρ2):(σ|3+6*σ|4+6*σ|5)
                     (ρ2,ρ1,ρ2,ρ2,ρ2):(3*σ|4+4*σ|5)
                     (ρ2,ρ1,ρ1,,ρ2,ρ1):(σ|3+7*σ|4+6*σ|5)
                     (ρ2,ρ1,ρ2,,ρ2,ρ1):(6*σ|3+18*σ|4+12*σ|5)
                     (ρ2,ρ1,ρ1,ρ1,ρ1,ρ1):(σ|6)
                     (ρ2,ρ1,ρ1,ρ1,ρ1,ρ2):(4*σ|5+5*σ|6)
                     (ρ2,ρ1,ρ1,ρ1,ρ2,ρ2):(3*σ|4+12*σ|5+10*σ|6)
                     (ρ2,ρ1,ρ1,ρ2,ρ2,ρ2):(3*σ|4+12*σ|5+10*σ|6)
                     (ρ2,ρ1,ρ2,ρ2,ρ2,ρ2):(4*σ|5+5*σ|6)
                     (ρ2,ρ1,ρ1,ρ1,,ρ2,ρ1):(3*σ|4+13*σ|5+10*σ|6)
                     (ρ2,ρ1,ρ1,ρ2,,ρ2,ρ1):
                                  (2*σ|3+24*σ|4+52*σ|5+30*σ|6)
                     (ρ2,ρ1,ρ2,ρ2,,ρ2,ρ1):
                                  (3*σ|3+27*σ|4+54*σ|5+30*σ|6)
                     (ρ2,ρ1,ρ2,,ρ2,ρ1,ρ1):
                                  (4*σ|3+21*σ|4+36*σ|5+20*σ|6);μ];
ρ
```

6.3 Rational expression transformations

The remaining transformation functions have as their domain the set J^f. They use the functions prod, sum, dif and invfac which have as their domain either the set Q or $Q \times Q$. Their range is the set Q. These are regarded as basic functions and detailed definition is not given. Details of such functions may be found in Davis [DAVIM580] or Markov [MARKA620]. In simple terms prod multiplies rational numbers, sum adds them and dif subtracts them. Function invfac evaluates the factorial of the argument and then inverts it.

Function trm transforms multiplicative expressions, trs transforms sum expressions and trd transforms difference expressions.

trm[σ] = - transform multiply

$$
\left[\begin{array}{l}
\sigma = \sigma_1 * 1 \rightarrow \sigma_1; \quad \sigma = 1 * \sigma_1 \rightarrow \sigma_1; \\
\sigma = \sigma_1 * 0 \rightarrow 0; \quad \sigma = 0 * \sigma_1 \rightarrow 0; \\
\sigma = -\sigma_1 * -\sigma_2 \rightarrow \text{scan}[\sigma_1 * \sigma_2; 0]; \\
\sigma = -\sigma_1 * \sigma_2 \rightarrow \text{scan}[-(\sigma_1 * \sigma_2); 0]; \\
\sigma = \sigma_1 * -\sigma_2 \rightarrow \text{scan}[-(\sigma_1 * \sigma_2); 0]; \\
\sigma = \sigma_1 * (\sigma_2 + \sigma_3) \rightarrow \text{scan}[\sigma_1 * \sigma_2 + \sigma_1 * \sigma_3; 0]; \\
\sigma = \sigma_1 * (\sigma_2 - \sigma_3) \rightarrow \text{scan}[\sigma_1 * \sigma_2 - \sigma_1 * \sigma_3; 0]; \\
\sigma = (\sigma_1 + \sigma_2) * \sigma_3 \rightarrow \text{scan}[\sigma_1 * \sigma_3 + \sigma_2 * \sigma_3; 0]; \\
\sigma = (\sigma_1 - \sigma_2) * \sigma_3 \rightarrow \text{scan}[\sigma_1 * \sigma_3 - \sigma_2 * \sigma_3; 0]; \\
\sigma = \sigma_1 * \sigma_2 \wedge \sigma_1 \varepsilon Q \wedge \sigma_2 \varepsilon Q \rightarrow \text{prod}[\sigma_1; \sigma_2]; \\
\sigma = \sigma_1 * \sigma_2 \wedge \text{ro}[\sigma_1; *] \overset{A}{>} \text{lo}[\sigma_2; *] \\
\qquad \rightarrow \text{scan}[\text{le}[\sigma_1; *]\text{lo}[\sigma_2; *]\text{ro}[\sigma_1; *]\text{re}[\sigma_2; *]; 0]; \\
\sigma
\end{array}\right]
$$

trs[σ] = - transform sum

$$\left[\begin{array}{l}
\sigma = \sigma_1+0 \rightarrow \sigma_1;\quad \sigma = 0+\sigma_1 \rightarrow \sigma_1; \\
\sigma = \sigma_1+\sigma_2 \wedge \sigma_1\varepsilon\mathcal{Q} \wedge \sigma_2\varepsilon\mathcal{Q} \rightarrow \mathrm{sum}[\sigma_1;\sigma_2]; \\
\sigma = \sigma_1+-\sigma_2 \rightarrow \mathrm{scan}[\sigma_1-\sigma_2;0]; \\
\sigma = -\sigma_1+\sigma_1 \rightarrow 0; \\
\sigma = \sigma_1+\sigma_1 \rightarrow \sigma_1*2; \\
\sigma = \sigma_1+\sigma_2 \wedge \mathrm{lts}[\sigma_2;\sigma_1] \rightarrow \sigma_2+\sigma_1; \\
\underline{\sigma}
\end{array}\right]$$

trd[σ] = - transform difference

$$\left[\begin{array}{l}
\sigma = \sigma_1-\sigma_1 \rightarrow 0; \\
\sigma = \sigma_1-0 \rightarrow \sigma_1;\quad \sigma = 0-\sigma_1 \rightarrow -\sigma_1; \\
\sigma = \sigma_1-\sigma_2 \wedge \sigma_1\varepsilon\mathcal{Q} \wedge \sigma_2\varepsilon\mathcal{Q} \rightarrow \mathrm{dif}[\sigma_1;\sigma_2]; \\
\sigma = \sigma_1-\sigma_2 \wedge \mathrm{lts}[\sigma_2;\sigma_1] \rightarrow -\sigma_2+\sigma_1; \\
\underline{\sigma}
\end{array}\right]$$

Combinatorial expressions are transformed by function trb and the function syp which generates a symbolic product.

syp[σ;μ_1;μ_2] = - symbolic product

$$\left[\begin{array}{l}
\mu_2 = \mu_1 \rightarrow \sigma; \\
\underline{\sigma}*\mathrm{syp}[(\sigma-\mu_2);\mu_1;\mathrm{sum}[\mu_2;1]]
\end{array}\right]$$

trb[σ] = - transform combination

$$[\sigma = \sigma_1|\sigma_2 \rightarrow \mathrm{syp}[\sigma_1;\sigma_2;1]*\mathrm{invfac}[\sigma_2]]$$

For convenience the transformation functions defined above are combined into a single function tf.

tf[τ;μ] = - transform

$$\left[\begin{array}{l} op[\tau] = . \rightarrow trp[\tau;\mu];\ \ op[\tau] = : \rightarrow tre[\tau;\mu]; \\ op[\tau] = , \rightarrow trc[\tau;\mu];\ \ op[\tau] = * \rightarrow trm[\tau]; \\ op[\tau] = + \rightarrow trs[\tau];\ \ op[\tau] = - \rightarrow trd[\tau]; \\ op[\tau] = | \rightarrow trb[\tau] \end{array}\right]$$

7. USE OF THE TRANSFORMATION RULES.

The transformation function tf is applied to expressions by the function scan. This has for its domain the set $E \times N'$ and for its range the set of canonic expressions.

scan[τ;μ] = - scan

$$\left[\begin{array}{l} \tau \epsilon RUL \rightarrow \tau; \\ tf[scan[red[lp'[\tau];modl[\tau;\mu]];modl[\tau;\mu]]op[\tau] \\ \quad scan[red[rp'[\tau;0];modr[\tau;\mu]];modr[\tau;\mu]]] \end{array}\right]$$

This function scans the left and right parts of the expression and then applies the function tf. This method is necessary since the transformations have been organised with the assumption that both the left and right parts of any expression to be transformed must be canonic. Such an assumption is not theoretically necessary but it does improve greatly the efficiency of the process by avoiding multiple scanning of the same expression.

It may be also observed that function red is applied to the left and right parts of the expression and that scan uses the functions modl and modr to generate modulii for scanning of the subexpressions.

This section completes the description of the collecting process.

8. COMPUTATIONAL TECHNIQUES.

8.1 Use of SLIP and PL1.

The computational techniques used belong to the fields of symbol manipulation and list processing. Background information is obtainable in a survey paper by Sammet [SAMMJ660] and a full description of the list processor is contained in Weizenbaum [WEIZJ630]. A version of Weizenbaum's symmetric list processor SLIP was implemented in the programming languages PL1 and Assembler.

The version has been called SYMPLIP to emphasise the differences. At the time that this work was done the list processing features of PL1 were not implemented. For this reason and because of the desire to minimise debugging time, the routines of SLIP were essentially transliterated into PL1 with deletion of only the most obviously obsolete features such as AND and LANORM. As well as this, the SLIP recursive features were deleted and where recursion was required as in LSTEQL, the recursive features of PL1 were used. The primitives were written in Assembler.

The major difficulty encountered in the implementation was due to the short word length of 32 bits in the S/360 and the fact that all addressing is with respect to bytes. Since the structure of SLIP would have been seriously affected by a decision to use three words per cell, a scheme was devised using a LAVS relocation constant, relative addressing within LAVS and storage of word addresses rather

than byte addresses. This approach retains the two word cell and permits the use of a 128 kilo-byte LAVS in a 256 kilo-byte store.

8.2 Representation of mathematical structures.

The terminal symbols (group expression generators or rational expression generators) are represented by two cell lists consisting of the header and a datum containing the symbol and type identification. Figure 1 shows the list representation for the group element a. The datum field is regarded as consisting of four characters. The first is unused. The second defines the type of datum.

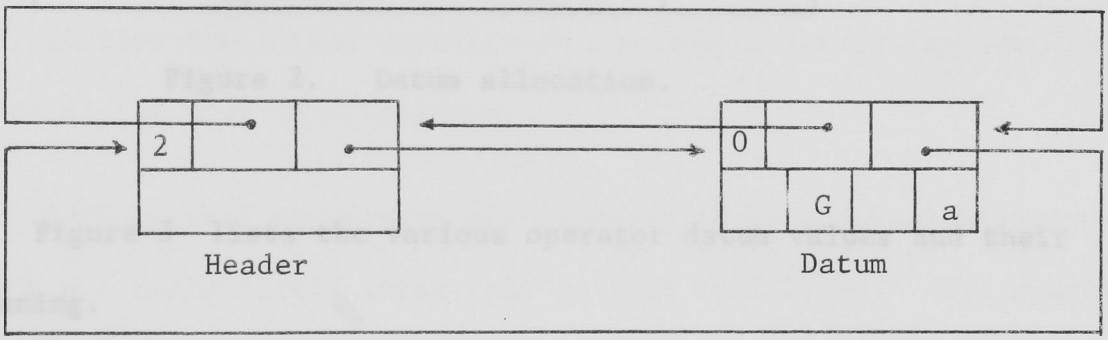
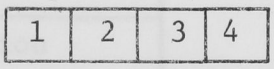


Figure 1. Representation of the generator a.

Figure 2 shows how the various character fields of the datum have been allocated.

Datum word character positions:



Character	Use	Value
1	Not used	
2	Operator designator	0
	Group expression generator	G
	Group identity	U
	Integer	I
	Rational expression generator	L
3	Special uses	
4	Datum value	

Figure 2. Datum allocation.

Figure 3 lists the various operator datum values and their meaning.

Datum values when character 2 = 0	Meaning
,	commutation
.	multiplication
:	exponentiation
'	group inversion
+	integer summation
/	integer division
|	combination
-	binary integer negation
U	unary " "
*	integer multiplication

Figure 3.

The multiple comma operators are not represented in Figure 3 because there is only one operation of commutation and the use of multiple commas is merely a device for suppression of parentheses without loss of structure in the expression. Parentheses do not appear in the list structure simply because, while they help to define structure in a linear string of symbols, the tree or list contains the structure of an expression inherently.

It will be noted in the description above and in the following text that the formal description of earlier sections is not in one-to-one correspondence with the current description. As mentioned in the introduction the formal description was simplified to improve clarity, often at the cost of efficiency.

Mathematical structures are represented by unary or binary trees which are respectively three cell or four cell lists. The operation a,b has the tree and list representation as shown in Figure 4.

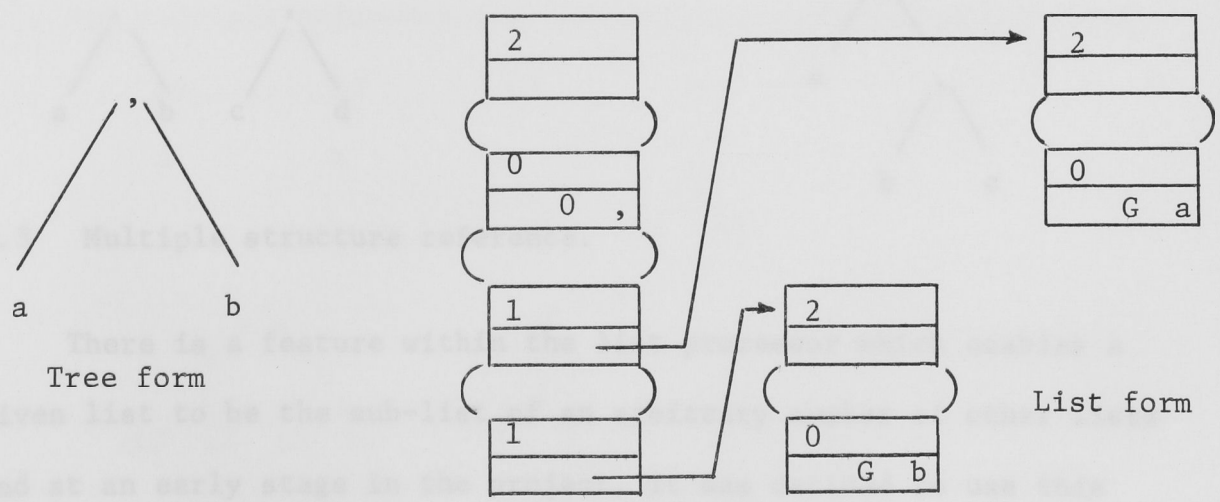
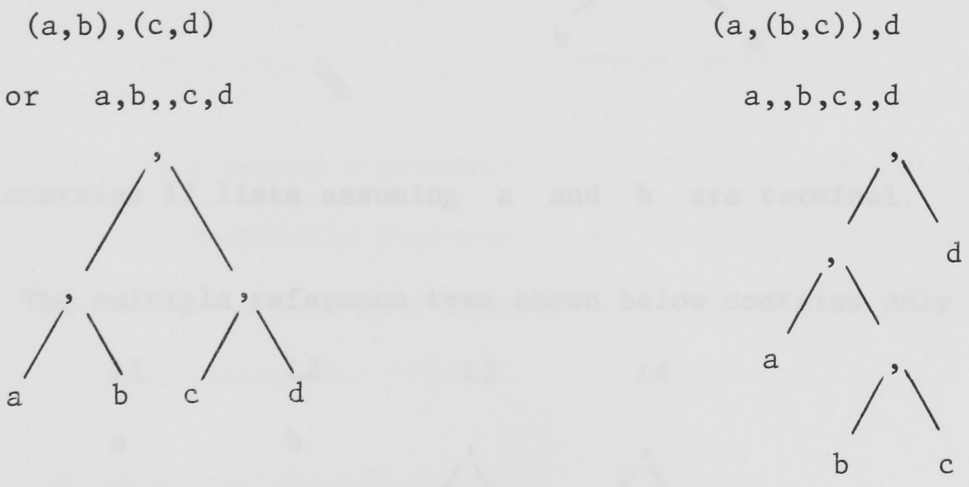


Figure 4.

This representation is general in that a or b themselves may be arbitrarily complex since the operation structure contains the names of the lists containing the operands.

While this choice of binary structure is perhaps the most obvious, it leads to prohibitive inefficiency when ordering products of commutators. After some reflection on this it is realised that since the associative law for multiplication holds, any binary tree representation of a product of more than two commutators will contain structure which is unrelated to the mathematical meaning of the expression. The means used to overcome this difficulty will be discussed later.

Consider the commutators (a,b),(c,d) and (a,(b,c)),d. These have the following tree structures.



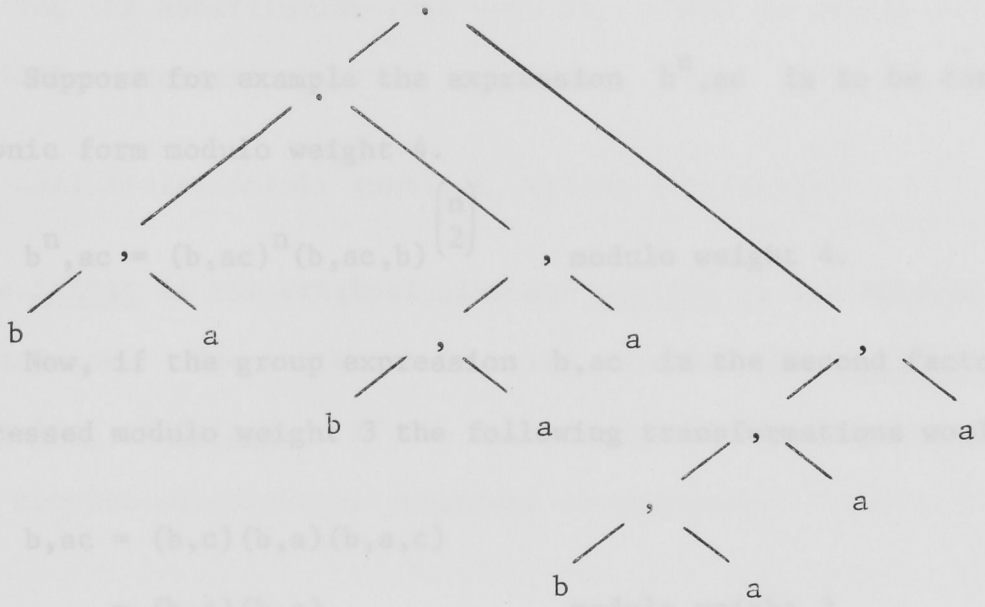
8.3 Multiple structure reference.

There is a feature within the list processor which enables a given list to be the sub-list of an arbitrary number of other lists and at an early stage in the project, it was decided to use this

feature to economise on the use of cells. Suppose, for example, that it is necessary to generate a list structure for the string

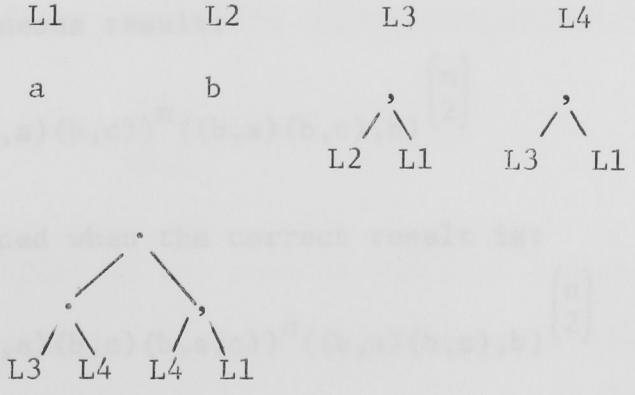
(b,a)(b,a,a)(b,a,a,a)

The single reference tree is as follows:



and contains 17 lists assuming a and b are terminal.

The multiple reference tree shown below contains only 7 lists



Multiple reference has a further advantage when coverting expressions to canonic form since the conversion need be performed only once for any given subexpression provided the resultant canonic form is not dependent on context, i.e. the surrounding mathematical structure. This can, unfortunately, occur in many cases and special precautions are required.

Suppose for example the expression b^n,ac is to be converted to canonic form modulo weight 4.

$$b^n,ac = (b,ac)^n(b,ac,b)^{\binom{n}{2}} \qquad \text{modulo weight 4.} \qquad \text{-- LEO}$$

Now, if the group expression b,ac in the second factor were processed modulo weight 3 the following transformations would occur:

$$b,ac = (b,c)(b,a)(b,a,c) \qquad \text{-- RPRDOUT}$$

$$= (b,a)(b,c) \qquad \text{modulo weight 3.}$$

Since this second expression replaces b,ac in the factor b,ac,b it automatically replaces b,ac in the first factor when in fact this factor should have been processed modulo weight 4. Thus the erroneous result:

$$((b,a)(b,c))^n((b,a)(b,c),b)^{\binom{n}{2}}$$

is produced when the correct result is:

$$((b,a)(b,c)(b,a,c))^n((b,a)(b,c),b)^{\binom{n}{2}}$$

8.4 Structure transformation

The decision to permit multiple list references complicates the task of replacing one list structure by another because it is not easily known just what lists refer to a list to be substituted. The difficulty is resolved by emptying the list to be replaced and building the substitution list onto it. This is simply performed by the SLIP statement:

```
call iralst(inlstr (outlist, mtlist (inlist)));
```

where inlist is the original list and outlist is the desired replacement. The statement is valid provided that outlist is not a sub-list of inlist. If the latter possibility exists a slightly more complex substitution procedure is required.

8.5 Transformation representation.

Each transformation is represented by a procedure usually of the form:

```
TRANS: PROCEDURE (STRUCTURE,MODULUS);

       . . .

       END;
```

STRUCTURE is the name of the list structure which is known to be in a suitable form for the transformation TRANS. The parameter MODULUS provides where necessary the contextual information necessary to guide the transformation algorithm. On exit from the procedure the structure to which STRUCTURE points has been appropriately transformed.

Because of the problem associated with multiply referenced lists and also in the interest of speed of computation, it is normal practice to avoid the generation of non-canonic commutators within the transformation procedure.

When for example, the transformation rule

ab,c = (a,c)(a,c,b)(b,c) is applied to ab,cd, modulo weight 4,

the first factor of the replacement expression generated is a,cd . Since this is itself non-canonic, it is immediately subjected to a further transformation of the type

$$a,bc = (a,c)(a,b)(a,b,c)$$

This transforms a,cd to (a,d)(a,c)(a,c,d). At this point the three commutators are still non-canonic since a<d and a<c . After further manipulation:

$$a,cd = (c,a)^{-1}(d,a)^{-1}(c,a,d)^{-1}$$

The second term of the first transformation may now be processed. It is obviously the first term commutated with b. That is

$$(c,a)^{-1}(d,a)^{-1}(c,a,d)^{-1},b$$

The first transformation is again called, effectively from within itself and eventually it realises that any commutator involving $(c,a,d)^{-1}$ and b will be of weight 4 and consequently equivalent to the identity. After further manipulation, this expression is reduced

to the canonic form $(c,a,b)^{-1}(d,a,b)^{-1}$. Note, of course, that only the main structure is replaced, the sub-structures, where applicable, remain attached to other super-structures.

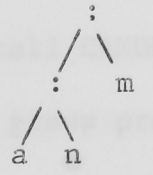
This is an efficient process because, on the assumption that both left and right parts of a structure are in canonic form, the depth of the non-canonic perturbation caused by the transformation is known in advance hence the canonic form can be restored more rapidly.

The manner in which this recursive behaviour is controlled will be explained below.

8.6 Pattern recognition or transformation selection.

Once a commutator expression is in the form of a list structure, the problem of pattern recognition and selection of transformation is trivial. The recognition process is centred in a procedure called CANONCL. Starting at the top of the tree this procedure makes a systematic attempt to classify the structure. The first basis for classification is the operator at the top of the tree. Observation of this operator permits immediate selection of one of the structure types described earlier in the text.

If, for example, CANONCL is presented with the structure $((a)^{n})^{m}$ which in tree form is



the operator at the top of the tree is ':' and we know that the structure belongs to the set of exponential structures. The next symbol to be checked is the operator of the left part of the structure and if as in the example, this is also ':' then the transformation PRODEXP is called.

If a match does not occur, it is valid to assume that the structure submitted to CANONCL is, in fact, canonic and it is returned unaltered. The simplicity of this recognition process results from the generality of the transformation rules and the structural simplicity of the list representation.

8.7 Structure scanning.

When an arbitrarily non-basic commutator expression is submitted for conversion to canonic form a recursive scanning procedure traces a path down to the terminal symbols and then begins the ascent. As each binary operation is encountered on the ascent, the structure is submitted to CANONCL which returns the corresponding canonic expression. Because this process of forming basic expressions was begun at the deepest level, it is always true that the left and right parts of the structure submitted to CANONCL are canonic and this in turn leads to simplification of the transformation procedures.

There is an exceptional situation in which the scanning procedure does not necessarily call CANONCL. This is the case in which the current operator is a group product and the operator of the super-

structure is also a group product. The purpose of this is to delay the process of product ordering until the largest possible string of product terms is available.

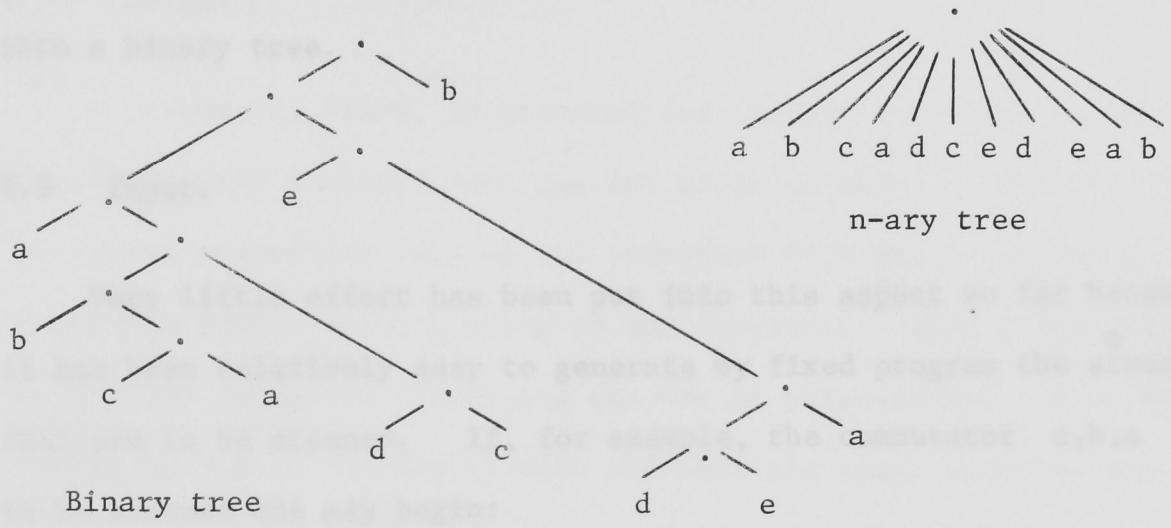
8.8 Ordering of products.

It will be recalled that the process of converting a commutator expression to canonic form consists of a sort of flotation process on the tree in which product operators are forced to assemble in a continuous block at the top of the structure followed by a thin layer of exponents (including inverses) followed by a block of commutation operators at the bottom. The upper layer of products will usually be very large and the time required for ordering them is some combinatorial power of the number of product operators in the string.

The original simple-minded appraoch to this product ordering problem was to treat product operations in exactly the same way as all other operations. In this way when the first binary product expression was detected by CANONCL the latter simply called the transformational procedure ORDPROD. This checked the order of right and left parts and if they were out of order it reordered them and introduced the additional commutator if necessary. While this is relatively simple in principle, it leads to a vast amount of back tracking and frequent repetitions of identical calculations.

To achieve tolerable efficiency in this vital aspect of the collecting process, it was necessary to assemble the largest possible

contiguous product string (i.e. no other imbedded operators) and transform it from an arbitrarily structured binary tree into a single simple n-ary tree. A diagram may help here:



Here the letters at the base of the tree may or may not be terminal but if they are non-terminal, they will be canonic and their uppermost operator is not a product.

The single node n-ary tree has the advantage that it can be scanned very rapidly and it is very easy to insert and delete components. When forming this simple list, various properties of the component expressions are evaluated and associated with the expressions so that they are evaluated only once during the ordering process. These quantities are the weight of the expression and its order number within weight. This latter number is so generated that for any two commutators a and b of equal weight if a<b then the order number of a is less than the order number of b.

With this method, the speed of ordering was improved by a factor of about 20 in some cases and an even greater improvement would be observed for larger expressions.

When ordering is complete, the n-ary tree is transformed back into a binary tree.

8.9 Input.

Very little effort has been put into this aspect so far because it has been relatively easy to generate by fixed program the structures that are to be scanned. If, for example, the commutator c,b,a is to be scanned one may begin:

```
a = list(9); call nxtlft(unspec('0G0a'),a);
b = list(9); call nxtlft(unspec('0G0b'),b);
c = list(9); call nxtlft(unspec('0G0c'),c);
ℓ = lg(',',lg(',',c,b),a);
```

whereupon ℓ contains the name of the commutator c,b,a .

Nevertheless, there are occasions where the input string is very long and the task of generating it in the above fashion is very laborious. For this reason, a procedure called STRNGET has been written for reading normal strings from cards and converting them into list structures. The routine is unsophisticated in that it assumes that the letters of the first half of the alphabet are group elements and those of the second half are literal representatives of integers.

Furthermore, although it does use multiple references to lists of the terminal symbols, it does not do this for non-terminal structures and the result can be a wasteful use of the list of available space.

8.10 Output.

A procedure, STRPN, is provided for transforming list structures to strings with standard notation and printing them. Both the input and output procedures call on the procedure PREC which returns a precedence number corresponding to any operator. This information is used for decisions concerning the use of parentheses. A further problem arises in connection with the multiple comma notation and it was found necessary to pre-scan the structure using the third character field of the structure datum word for recording the number of commas where applicable. This information is then readily accessible to the print routine which is a relatively simple recursive procedure.

While the datum word leaves adequate scope for a more extensive symbol range, input-output considerations lead to undesirable restrictions. Firstly, it is preferable not to use multi-character identifiers for letters since this makes expressions too long and therefore harder to read and also the convention of implicit multiplication must be dropped. Secondly, it has proved to be essential to have in excess of 30 or 40 symbols for group elements and exponents.

The ideal of course, is to be able to write expressions very similar to the following:

$$\begin{pmatrix} \alpha_1 & \beta_1 & \gamma_1 \\ a_1 & b_1 & c_1 \end{pmatrix}, \begin{pmatrix} \alpha_2 & \beta_2 & \gamma_2 \\ a_2 & b_2 & c_2 \end{pmatrix}$$

and graphic display with electrostatic printer will be necessary before this difficulty is overcome.

9. EFFECTIVENESS OF THE IMPLEMENTATION.

An encouraging degree of success has been achieved with the automatic collecting process. One of the first expressions to be collected was $(ab)^4$ modulo weight 7 and this computation took 13 minutes of machine time. The computation, which was checked using the formulae of Krause [KRAUE640], has not been repeated by hand but it is expected that it would take many hours.

An assortment of computations on expressions of the form:

$$a^{\alpha_1}b^{\beta_1}(b,a)^{\gamma_1},a^{\alpha_2}b^{\beta_2}(b,a)^{\gamma_2}$$

using literal exponents have been performed and the results checked by hand. In these cases the machine has proved to be only 5 or 10 times faster than the human calculator but far less subject to error. It is expected that the ratio of machine speed would increase with problem complexity because the machine has a more efficient technique for modifying expressions and in a relative sense the longer the expression the more effective the technique.

Much of the inefficiency of this automatic collecting process is due to the SLIP=PL1 combination. Even the simplest list processing operations are executed by a long chain of functions which in many cases could be replaced by a small number of in-line instructions, taking less space than the function call.

However, there is still room for significant improvement of the algorithm. This is often realised when checking through the

intermediate printout of a computation. In several cases, it can be observed that the machine achieves a result by a very devious path when the result is immediately obvious to the human calculator. In all such cases, the means for improving the machine algorithm have been quite simple but each improvement causes an increase in the size of the program and consequently less workspace.

An interesting example is that of evaluating

$$((b,a,b,,b,a)(b,a,b,b,b))^3 \qquad \text{modulo weight 6}$$

To the human operator the result is immediate:

$$(b,a,b,,b,a)^3(b,a,b,b,b)^3$$

The computer procedures however, proceed as follows:

$$\begin{aligned}
&((b,a,b,,b,a)(b,a,b,b,b))^3 && \text{modulo weight 6}\\
&= (b,a,b,,b,a)(b,a,b,b,b)(b,a,b,,b,a)(b,a,b,b,b)\\
&\quad (b,a,b,,b,a)(b,a,b,b,b)\\
&= (b,a,b,,b,a)^2(b,a,b,b,b)^2(b,a,b,,b,a)(b,a,b,b,b)\\
&= (b,a,b,,b,a)^3(b,a,b,b,b)^3
\end{aligned}$$

In cases where the value of the exponent is large, this is a very slow process.

There is another problem for the machine algorithm in its competition with the human algorithm. This is the problem of context dependent transformations which were discussed in Section 5. While it has been possible to build a very small number of these into the

machine algorithm the expreienced human calculator has a large range of them at his disposal and can rapidly determine just when they should be applied.

For example, the commutator expression ab,cd modulo weight 3 is (a,c)(a,d)(b,c)(b,d). This is a sort of macro transformation which is applicable when the expression is already of maximum weight.

This example may be partially generalised as shown in the following transformation:

$$\left(\prod_{i=1}^{m} a_i\right),\left(\prod_{j=1}^{n} b_j\right) = \prod_{i=1}^{m}\left(\prod_{j=1}^{n} (a_i,b_j)\right) \qquad \text{modulo weight 3.}$$

Here the normally recursive methods for breaking down an expression are replaced by an iterative method, and instead of being able to select the required transformation on the basis of a couple of levels of operators it is necessary to consider an arbitrary number of levels.

The range of such transformations is limited only by the experience and memory of the human calculator. Machine algorithms of this type would in some of the calculations performed give rise to speed increases of 10 or more.

In this approach lies the greatest promise for an efficient algorithm. But the approach must be pursued fundamentally and systematically rather than heuristically. The transformations used in this text were originally chosen as the smallest and simplest set amenable to inductive proof and devoid of considerations of algorithmic efficiency. A completely new set of transformations should be developed with the latter aim in mind.

10. TERMS, SYMBOLS AND NOTATIONS USED IN THE TEXT.

10.1 Commonly used symbols and notations.

$\wedge$	propositional connective and
$\vee$	propositional connective or
$\sim$	propositional connective not
=>	implication (note different usage in Section 10.4)
<=>	equivalence
$\exists$	there exists
$\{x \mid P(x)\}$	the set of elements x with property P
$A \cup B$	the union of sets A and B
$A \cap B$	the intersection of sets A and B
$\bigcup_{i=m}^{i=n} A_i$	the union of a finite family of sets
$a \in A$	a is a member of the set A
$a \notin A$	a is not a member of the set A
$A \subseteq B$	A is a subset of B
$\binom{n}{r}$	$= n!/((n-r)!r!)$
x,y	the element $x^{-1} y^{-1} xy$ and known as a commutator
∞	infinity
Π	product function
function	a function is a triple (D,R,f) where D and R are non-empty sets and to each element x in D there corresponds an element denoted f(x) in R. D is called the domain and R the range of the function.
predicate	a function whose range is the set {T,F}

10.2 Symbols and notations introduced in the text.

Symbol	Explanation	Page
Λ	the null word	21
[	left meta-bracket	33, 89
]	right meta-bracket	33, 89
ϕ^{ℓ}	the function lr[ϕ]	22
ϕ^{r}	the function rs[ϕ]	22
T	the truth value true	24, 34, 89
F	the truth value false	24, 34, 89
<	left diamond bracket	8
>	right diamond bracket	8
::=	becomes	8
|	separator for alternatives of a rule	8
<	integer or rational relation less than	34
>	integer or rational relation greater than	25, 34
=	integer or rational relation	34, 35
$\underset{<}{A}$	alphabetic order less than	35, 38
$\underset{>}{A}$	alphabetic order greater than	35
<•	relation in precedence grammar	18, 88
•>	relation in precedence grammar	18, 88
$\doteq$	relation in precedence grammar	18, 88
;	meta-delimiter	89
{<x>}	the set of all derivations of meta-variable <x>	34
a,...,k	group expression generators	3, 10
0,...,9	digits	3, 10
ℓ,...,z	rational expression generators	3, 9

ν	group identity	3, 10
+	rational summation operator	3, 10
-	rational difference operator	3, 10
*	rational product operator	3, 10
|	rational combination operator	3, 10
/	rational division operator	3, 10
:	group exponent operator	3, 10
.	group product operator	4, 16
,	group commutator operator	3, 4, 11
$,_\alpha$	subscripted commutator operator	4, 5, 11, 16
(	left parenthesis	3, 10
)	right parenthesis	3, 10
$\frown$	digit concatenation operator	4, 16

$\mathcal{B}$	{<rational expression generator>}	34, 39
E	$\{G \cup J\}$	17
$\hat{E}$	set of elements of E truncated on the left or right	20
E'	$\{G' \cup J'\}$	17
E^f	fully parenthesised subset of E	29
G	the internal grammar	17
G'	the input/output grammar	11, 17
G	{<group expression>}	5, 6, 17
G'	input/output equivalent of G	3, 11, 17
G^f	fully parenthesised subset of G	6, 29
J	{<rational expression>}	5, 17

J'	input/output equivalent of J	3, 10, 17
L	{<group expression generator>}	34, 37
N	{<integer>}	20, 22
N'	{<integer>} $\cup$ {∞}	34, 40
O	{:;/;|;+;-;.;,;^;*}	34, 36
Q	{<rational number>}	34, 36
R	$\{B \cup Q\}$	34, 36
S	$\{R \cup L \cup O\}$ and symbols of the form $',_{\alpha}'$	5, 17
S'	$\{R \cup L \cup O\} \setminus \{.;\frown\}$	3, 4, 17
S'^*	the semigroup on S'	3, 17
S^*	the semigroup on S	5, 17
T	{T;F}	23, 34

γ	variable with domain G	3
λ	variable with domain J	3
ρ	variable with domain G^f	40
σ	variable with domain J^f	36
τ	variable with domain E^f	35
ϕ	variable with domain $\hat{E}$ or E	20, 24
X	variable with domain O	37
ψ	variable with domain $\hat{E}$ or E	21
μ	variable with domain N' or N	20, 22
α	variable with domain N	4, 11

10.3 Functions and terms introduced in the text.

prod	product	54
rat	rational	36
re	right expression	37
red	reduce	47
ro	right operand	37
rp	right part	35
rp'	right part	25
rpr	right product	39
rr	right remainder	21
rs	right symbol	21
scan	scan expression	57
skl	skip level	23
sum	summation	54
syp	symbolic product	55
tf	transform	56
tr	trailing part	41
trb	transform combinatorial	55
trc	transform commutator	51
trd	transform difference	55
tre	transform exponent	53
trm	transform rational product	54
trp	transform group product	50
trs	transform sum	55
wt	weight	40

10.4 Context free grammars and languages.

alphabet: a finite non empty set

sentence on an alphabet: a finite sequence of elements of the alphabet

word: a sentence

string: a sentence

Λ the sentence of length zero

S^* the set of all words over (or on) an alphabet S

context-free grammar: a 4-tuple $G = (V,S,P,s)$ where

V is an alphabet

$S \subseteq V$ is an alphabet

P is a finite set of ordered pairs

(u,v) with $u \in \{V-S\} - \{\Lambda\}$ and

$v \in V^*$

$s \in V-S$

The alphabet S is the set of symbols which will appear in words or sentences of the language (to be defined). These symbols are sometimes called terminal symbols.

The alphabet $V-S$ is a set of symbols which are used in defining the grammar of the language. These are sometimes called non-terminal symbols or meta-symbols.

P is a set of rules for transforming words on the meta-symbols into words on the full alphabet. Instead of being written in the form (u,v) these rules may be written $u \to v$.

s is the distinguished meta-symbol and usually will stand for 'a sentence' or 'a program'.

$w \Rightarrow y$ or w generates y for $w,y \in V^*$ if there exist z_1, z_2, u and v and $w = z_1 u z_2$ and $y = z_1 v z_2$ and $(u,v) \in P$.

$w \overset{*}{\Rightarrow} y$ if either $w = y$ or there exist $w_0, w_1, \ldots, w_r$ such that $w_0 = w$, $w_r = y$ and $w_i \Rightarrow w_{i+1}$. $w \overset{*}{\Rightarrow} y$ is called a generation or derivation.

context free language:

If G is a context free grammar then the subset L(G) of S^* defined by $L(G) = \{x \in S^* \mid s \overset{*}{\Rightarrow} x\}$ is called a context-free language

Informally this means that the language is made up of all those words in S^* which can be derived from the meta-variable s or the set of words which can be parsed in terms of the definition of a grammatical sentence.

10.5 Ambiguity of formal grammars.

A leftmost derivation is a derivation

$$w_0 \Rightarrow \ldots \Rightarrow w_r$$

such that for each i, $w_i = u_i v_i y_i$ and $w_{i+1} = u_i z_i y_i$ with $(v_i, z_i) \in P$ and $u_i \in S^*$.

A grammar G is ambiguous if there is some word in L(G) which can be generated by two different leftmost derivations.

10.6 Operator grammars.

An operator grammar is a context-free grammar such that no member of P takes the form (U,xU_1U_2y) where $U,U_1,U_2 \in V-S$ and x,y are arbitrary strings.

10.7 Precedence grammars.

A precedence grammar is an operator grammar for which no more than one of the three relations $<\cdot$, $\doteq$ or $\cdot>$ hold between ordered pairs of terminal symbols. These relations are defined below.

$T_1 \doteq T_2$ if $T_1,T_2 \in S$ and there is a rule (U,xT_1T_2y) or $(U,xT_1U_1T_2y)$ where $U_1 \in V-S$.

$T_1 \cdot> T_2$ if there is a rule (U,xU_1T_2y) and a derivation $U_1 \Rightarrow z$ where $U_1 \in V-S$ and T_1 is the rightmost terminal symbol of z.

$T_1 <\cdot T_2$ if there is a rule (U,xT_1U_1y) and a derivation $U_1 \Rightarrow z$ where $U_1 \in V-S$ and T_2 is the leftmost terminal symbol of z.

These relations are called precedence relations.

If a context-free grammar is ambiguous it is not a precedence grammar. This follows from the two definitions and the parsing algorithm whose uniqueness is dependent on the uniqueness of the precedence relations.

10.8 Conditional expressions.

A conditional expression has the form

$$[p_1 \rightarrow e_1;\ p_2 \rightarrow e_2; \ldots ; p_n \rightarrow e_n] \quad \text{or}$$

$$[p_1 \rightarrow e_1;\ p_2 \rightarrow e_2; \ldots ; p_n \rightarrow e_n ; e_{n+1}]$$

where the p_i are propositional expressions or predicates which have the value T or F (standing for true or false), and the e_i are expressions.

The predicates may themselves be conditional expressions as also may the expressions. In the text the expressions are either group expressions, rational expressions, integer expressions or predicates.

The conditional expressions above have the value of the first e_i such that p_i is the leftmost predicate having the value T. In the case of the second conditional expression if all p_i have the value F then it takes the value e_{n+1}.

If there is an undefined p_i and no true p_j to the left of it then the expression is undefined.

If p_i is the leftmost true predicate and e_i is undefined then the expression is undefined.

If in the case of the first conditional expression there is no true predicate then the conditional expression is undefined.

These conditional expressions are used to define functions which are called conditional functions.

```
BASIC:  PROC(COM)          BIT(1)    RECURSIVE;                          BASIC    0
        /* RETURNS '1'B IFF COM IS BASIC */                               BASIC    1
        DCL  COM  FIXED BIN(31,0);                                        BASIC    2
        DCL  BASIC  ENTRY(FIXED BIN(31,0))RETURNS(BIT(1));                BASIC    3
        DCL  (                                                            BASIC    4
             WT,BOT,CONT                                                  BASIC    5
             )ENTRY(FIXED BIN(31,0))RETURNS(FIXED BIN(31,0));             BASIC    6
        DCL  (                                                            BASIC    7
             LT,EQ                                                        BASIC    8
             )ENTRY(FIXED BIN(31,0),FIXED BIN(31,0))                      BASIC    9
             RETURNS(BIT(1));                                             BASIC   10
        DCL (                                                             BASIC   11
             L,R                                                          BASIC   12
             )FIXED BIN(31,0);                                            BASIC   13
        DCL  (                                                            BASIC   14
             MADNTP                                                       BASIC   15
             )ENTRY(FIXED BIN(31,0),FIXED BIN(31,0))                      BASIC   16
             RETURNS(FIXED BIN(31,0));                                    BASIC   17
        DCL NTERM ENTRY(FIXED BIN(31,0))RETURNS(BIT(1));                  BASIC   18
        IF ¬ NTERM(COM) THEN RETURN('1'B);                                BASIC   19
        L=CONT(MADNTP(COM,2)+1); R=BOT(COM);                              BASIC   20
        IF NTERM(L) THEN RETURN (BASIC(L) & BASIC(R) & LT(R,L)            BASIC   21
                                 & (LT(BOT(L),R) | EQ(BOT(L),R)));        BASIC   22
                    ELSE RETURN (BASIC(L) & BASIC(R) & LT(R,L));          BASIC   23
        END BASIC;                                                        BASIC   24


CANCEL: PROC(STR);                                                        CANCEL   0
        /* A'A OR AA'-> 1  */                                             CANCEL   1
        DCL (                                                             CANCEL   2
```

```
        CAL,STR,A                                                          CANCEL   3
        )    FIXED BIN(31,0);                                              CANCEL   4
      DCL (                                                                CANCEL   5
        NXTLFT,INLSTR                                                      CANCEL   6
        )ENTRY(FIXED BIN(31,0),FIXED BIN(31,0))                            CANCEL   7
        RETURNS(FIXED BIN(31,0));                                          CANCEL   8
      DCL LIST ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));            CANCEL   9
      DCL (                                                                CANCEL  10
        MTLIST                                                             CANCEL  11
        )ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));                  CANCEL  12
      A=LIST(9);                                                           CANCEL  13
      CAL=NXTLFT(UNSPEC('0U00'),A);                                        CANCEL  14
      CALL IRALST(INLSTR(A,MTLIST(STR)));                                  CANCEL  15
      END CANCEL;                                                          CANCEL  16


ADVLEL: PROC(LR,A)          FIXED BIN(31,0);                               ADVLEL   0
      DCL (LR,A)            FIXED BIN(31,0);                               ADVLEL   1
      DCL REED   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));          ADVLEL   2
      DCL ADVLL ENTRY(FIXED BIN(31,0),FIXED BIN(31,0),                     ADVLEL   3
      FIXED BIN(31,0))RETURNS(FIXED BIN(31,0));                            ADVLEL   4
      A=ADVLL(LR,0,0);                                                     ADVLEL   5
      IF A=0 THEN RETURN(REED(LR));                                        ADVLEL   6
      RETURN (A);                                                          ADVLEL   7
      END ADVLEL;                                                          ADVLEL   8


ADVLER: PROC(LR,A)          FIXED BIN(31,0);                               ADVLER   0
      DCL (LR,A)            FIXED BIN(31,0);                               ADVLER   1
      DCL REED   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));          ADVLER   2
      DCL ADVLR ENTRY(FIXED BIN(31,0),FIXED BIN(31,0),                     ADVLER   3
      FIXED BIN(31,0))RETURNS(FIXED BIN(31,0));                            ADVLER   4
```

```
        A=ADVLR(LR,0,0);                                                    ADVLER    5
        IF A=0 THEN RETURN(REED(LR));                                       ADVLER    6
        RETURN (A);                                                         ADVLER    7
        END ADVLER;                                                         ADVLER    8


ADVLL:  PROC(LR,J,K)          FIXED BIN(31,0);                              ADVLL     0
        DCL(LR,J,K)           FIXED BIN(31,0);                              ADVLL     1
        DCL CONT   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));         ADVLL     2
        DCL ID     ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));         ADVLL     3
        DCL LNKL   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));         ADVLL     4
        DCL SETDIR ENTRY(FIXED BIN(31,0),FIXED BIN(31,0),                   ADVLL     5
                         FIXED BIN(31,0),FIXED BIN(31,0))                   ADVLL     6
                   RETURNS(FIXED BIN(31,0));                                ADVLL     7
        DCL STRIND ENTRY(FIXED BIN(31,0),FIXED BIN(31,0))                   ADVLL     8
                   RETURNS(FIXED BIN(31,0));                                ADVLL     9
        DCL(CLR,LK,CAND,LLV) FIXED BIN(31,0) STATIC;                        ADVLL    10
        DCL CAL FIXED BIN(31,0) STATIC;                                     ADVLL    11
        CLR=CONT(LR);                                                       ADVLL    12
   L5:  LK=LNKL(CONT(LNKL(CLR)));                                           ADVLL    13
        CAND=CONT(LK);                                                      ADVLL    14
        CALL SETDIR(-1,LK,-1,CLR);                                          ADVLL    15
        IF ID(CAND)=2 THEN GOTO L2;                                         ADVLL    16
        IF ID(CAND)¬=J & ID(CAND)¬=K THEN GOTO L5;                          ADVLL    17
        LLV=0; GOTO L6;                                                     ADVLL    18
   L2:  LLV=-1;                                                             ADVLL    19
   L6:  CAL= STRIND(CLR,LR);                                                ADVLL    20
        RETURN(LLV);                                                        ADVLL    21
        END ADVLL;                                                          ADVLL    22


ADVLNL: PROC(LR,A)            FIXED BIN(31,0);                              ADVLNL   0
```

```
        DCL (LR,A)          FIXED BIN(31,0);                                    ADVLNL    1
        DCL REED   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));             ADVLNL    2
        DCL ADVLL ENTRY(FIXED BIN(31,0),FIXED BIN(31,0),                        ADVLNL    3
        FIXED BIN(31,0))RETURNS(FIXED BIN(31,0));                               ADVLNL    4
        A=ADVLL(LR,1,1);                                                        ADVLNL    5
        IF A=0 THEN RETURN(REED(LR));                                           ADVLNL    6
        RETURN (A);                                                             ADVLNL    7
        END ADVLNL;                                                             ADVLNL    8


ADVLNR: PROC(LR,A)          FIXED BIN(31,0);                                    ADVLNR    0
        DCL (LR,A)          FIXED BIN(31,0);                                    ADVLNR    1
        DCL REED   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));             ADVLNR    2
        DCL ADVLR ENTRY(FIXED BIN(31,0),FIXED BIN(31,0),                        ADVLNR    3
        FIXED BIN(31,0))RETURNS(FIXED BIN(31,0));                               ADVLNR    4
        A=ADVLR(LR,1,1);                                                        ADVLNR    5
        IF A=0 THEN RETURN(REED(LR));                                           ADVLNR    6
        RETURN (A);                                                             ADVLNR    7
        END ADVLNR;                                                             ADVLNR    8


ADVLR:  PROC(LR,J,K)        FIXED BIN(31,0);                                    ADVLR     0
        DCL(LR,J,K)         FIXED BIN(31,0);                                    ADVLR     1
        DCL CONT   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));             ADVLR     2
        DCL ID     ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));             ADVLR     3
        DCL LNKL   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));             ADVLR     4
        DCL LNKR   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));             ADVLR     5
        DCL SETDIR ENTRY(FIXED BIN(31,0),FIXED BIN(31,0),                       ADVLR     6
                   FIXED BIN(31,0),FIXED BIN(31,0))                             ADVLR     7
                   RETURNS(FIXED BIN(31,0));                                    ADVLR     8
        DCL STRIND ENTRY(FIXED BIN(31,0),FIXED BIN(31,0))                       ADVLR     9
                   RETURNS(FIXED BIN(31,0));                                    ADVLR    10
```

```
        DCL(CLR,LK,CAND,LRV) FIXED BIN(31,0) STATIC;                    ADVLR   11
        DCL CAL FIXED BIN(31,0) STATIC;                                 ADVLR   12
        CLR=CONT(LR);                                                   ADVLR   13
    L5: LK=LNKR(CONT(LNKL(CLR)));                                       ADVLR   14
        CAND=CONT(LK);                                                  ADVLR   15
        CALL SETDIR(-1,LK,-1,CLR);                                      ADVLR   16
        IF ID(CAND)=2 THEN GOTO L2;                                     ADVLR   17
        IF ID(CAND)¬=J & ID(CAND)¬=K THEN GOTO L5;                      ADVLR   18
        LRV=0; GOTO L6;                                                 ADVLR   19
    L2: LRV=-1;                                                         ADVLR   20
    L6: CAL= STRIND(CLR,LR);                                            ADVLR   21
        RETURN(LRV);                                                    ADVLR   22
        END ADVLR;                                                      ADVLR   23


ADVLWL: PROC(LR,A)          FIXED BIN(31,0);                            ADVLWL   0
        DCL (LR,A)          FIXED BIN(31,0);                            ADVLWL   1
        DCL REED   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));     ADVLWL   2
        DCL ADVLL ENTRY(FIXED BIN(31,0),FIXED BIN(31,0),                ADVLWL   3
        FIXED BIN(31,0))RETURNS(FIXED BIN(31,0));                       ADVLWL   4
        A=ADVLL(LR,1,0);                                                ADVLWL   5
        IF A=0 THEN RETURN(REED(LR));                                   ADVLWL   6
        RETURN (A);                                                     ADVLWL   7
        END ADVLWL;                                                     ADVLWL   8


ADVLWR: PROC(LR,A)          FIXED BIN(31,0);                            ADVLWR   0
        DCL (LR,A)          FIXED BIN(31,0);                            ADVLWR   1
        DCL REED   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));     ADVLWR   2
        DCL ADVLR ENTRY(FIXED BIN(31,0),FIXED BIN(31,0),                ADVLWR   3
        FIXED BIN(31,0))RETURNS(FIXED BIN(31,0));                       ADVLWR   4
        A=ADVLR(LR,1,0);                                                ADVLWR   5
```

```
        IF A=0 THEN RETURN(REED(LR));                                          ADVLWR   6
        RETURN (A);                                                            ADVLWR   7
        END ADVLWR;                                                            ADVLWR   8


ADVSEL: PROC(LR,A)          FIXED BIN(31,0);                                   ADVSEL   0
        DCL (LR,A)          FIXED BIN(31,0);                                   ADVSEL   1
        DCL REED   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));            ADVSEL   2
        DCL ADVSL ENTRY(FIXED BIN(31,0),FIXED BIN(31,0),                       ADVSEL   3
        FIXED BIN(31,0))RETURNS(FIXED BIN(31,0));                              ADVSEL   4
        A=ADVSL(LR,0,0);                                                       ADVSEL   5
        IF A=0 THEN RETURN(REED(LR));                                          ADVSEL   6
        RETURN (A);                                                            ADVSEL   7
        END ADVSEL;                                                            ADVSEL   8


ADVSER: PROC(LR,A)          FIXED BIN(31,0);                                   ADVSER   0
        DCL (LR,A)          FIXED BIN(31,0);                                   ADVSER   1
        DCL REED   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));            ADVSER   2
        DCL ADVSR ENTRY(FIXED BIN(31,0),FIXED BIN(31,0),                       ADVSER   3
        FIXED BIN(31,0))RETURNS(FIXED BIN(31,0));                              ADVSER   4
        A=ADVSR(LR,0,0);                                                       ADVSER   5
        IF A=0 THEN RETURN(REED(LR));                                          ADVSER   6
        RETURN (A);                                                            ADVSER   7
        END ADVSER;                                                            ADVSER   8


ADVSL:  PROC(L,J,K)          FIXED BIN(31,0);                                  ADVSL    0
        DCL (L,J,K)          FIXED BIN(31,0);                                  ADVSL    1
        DCL (R,LCP,M,ADVSA,CAND,LK)      FIXED BIN(31,0) STATIC;             ADVSL    2
        DCL CONT   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));            ADVSL    3
```

```
      DCL INHALT ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));          ADVSL   4
      DCL ID     ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));          ADVSL   5
      DCL LNKL   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));          ADVSL   6
      DCL LNKR   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));          ADVSL   7
      DCL SETDIR ENTRY(FIXED BIN(31,0),FIXED BIN(31,0),                    ADVSL   8
                       FIXED BIN(31,0),FIXED BIN(31,0))                    ADVSL   9
                 RETURNS(FIXED BIN(31,0));                                 ADVSL  10
      DCL SETIND ENTRY(FIXED BIN(31,0),FIXED BIN(31,0),                    ADVSL  11
                       FIXED BIN(31,0),FIXED BIN(31,0))                    ADVSL  12
                 RETURNS(FIXED BIN(31,0));                                 ADVSL  13
      DCL STRIND ENTRY(FIXED BIN(31,0),FIXED BIN(31,0))                    ADVSL  14
                 RETURNS(FIXED BIN(31,0));                                 ADVSL  15
      DCL NUCELL ENTRY RETURNS(FIXED BIN(31,0));                           ADVSL  16
      DCL LCNTR  ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));          ADVSL  17
      DCL RCELL  ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));          ADVSL  18
      DCL CAL FIXED BIN(31,0) STATIC;                                      ADVSL  19
      R=CONT(L); CAND=CONT(LNKL(R));                                       ADVSL  20
      IF ID(CAND)=1 THEN GOTO L6;                                          ADVSL  21
L1:   LCP=LNKL(CAND); CALL SETDIR(-1,LCP,-1,R);                            ADVSL  22
      CAND=CONT(LCP);                                                      ADVSL  23
      IF ID(CAND)=2 THEN GOTO L4;                                          ADVSL  24
      IF ID(CAND)=J | ID(CAND)=K THEN GOTO L8;                             ADVSL  25
      IF ID(CAND)¬=1 THEN GOTO L1;                                         ADVSL  26
L6:   M=NUCELL; CAL= STRIND(R,M); CAL= STRIND(CONT(L+1),M+1);              ADVSL  27
      CALL SETIND(-1,LNKR(CONT(LNKL(R)+1)),LCNTR(L)+1,L+1);                ADVSL  28
      CALL SETDIR(-1,-1,M,R);                                              ADVSL  29
      CAND=CONT(INHALT(LNKL(R)+1));                                        ADVSL  30
      GOTO L1;                                                             ADVSL  31
L4:   IF LCNTR(L)¬=0 THEN GOTO L9;                                         ADVSL  32
      ADVSA=-1; GOTO L12;                                                  ADVSL  33
L9:   LK=LNKR(R); R=CONT(LK); CAL= STRIND(CONT(LK+1),L+1);                 ADVSL  34
      CAND=CONT(LNKL(R)); CALL RCELL(LK);                                  ADVSL  35
      GOTO L1;                                                             ADVSL  36
L8:   ADVSA=0;                                                             ADVSL  37
```

```
    L12: CAL= STRIND(R,L);                                                    ADVSL   38
         RETURN(ADVSA);                                                       ADVSL   39
         END ADVSL;                                                           ADVSL   40


ADVSNL:  PROC(LR,A)          FIXED BIN(31,0);                                 ADVSNL   0
         DCL (LR,A)          FIXED BIN(31,0);                                 ADVSNL   1
         DCL REED   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));          ADVSNL   2
         DCL ADVSL ENTRY(FIXED BIN(31,0),FIXED BIN(31,0),                     ADVSNL   3
         FIXED BIN(31,0))RETURNS(FIXED BIN(31,0));                            ADVSNL   4
         A=ADVSL(LR,1,1);                                                     ADVSNL   5
         IF A=0 THEN RETURN(REED(LR));                                        ADVSNL   6
         RETURN (A);                                                          ADVSNL   7
         END  ADVSNL;                                                         ADVSNL   8


ADVSNR:  PROC(LR,A)          FIXED BIN(31,0);                                 ADVSNR   0
         DCL (LR,A)          FIXED BIN(31,0);                                 ADVSNR   1
         DCL REED   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));          ADVSNR   2
         DCL ADVSR ENTRY(FIXED BIN(31,0),FIXED BIN(31,0),                     ADVSNR   3
         FIXED BIN(31,0))RETURNS(FIXED BIN(31,0));                            ADVSNR   4
         A=ADVSR(LR,1,1);                                                     ADVSNR   5
         IF A=0 THEN RETURN(REED(LR));                                        ADVSNR   6
         RETURN (A);                                                          ADVSNR   7
         END ADVSNR;                                                          ADVSNR   8


ADVSR:   PROC(L,J,K)         FIXED BIN(31,0);                                 ADVSR    0
         DCL (L,J,K)         FIXED BIN(31,0);                                 ADVSR    1
         DCL (R,LCP,M,ADVSA,CAND,LK)      FIXED BIN(31,0) STATIC;             ADVSR    2
         DCL CONT   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));          ADVSR    3
```

```
      DCL INHALT ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));          ADVSR   4
      DCL ID     ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));          ADVSR   5
      DCL LNKL   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));          ADVSR   6
      DCL LNKR   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));          ADVSR   7
      DCL SETDIR ENTRY(FIXED BIN(31,0),FIXED BIN(31,0),                    ADVSR   8
                       FIXED BIN(31,0),FIXED BIN(31,0))                    ADVSR   9
                 RETURNS(FIXED BIN(31,0));                                 ADVSR  10
      DCL SETIND ENTRY(FIXED BIN(31,0),FIXED BIN(31,0),                    ADVSR  11
                       FIXED BIN(31,0),FIXED BIN(31,0))                    ADVSR  12
                 RETURNS(FIXED BIN(31,0));                                 ADVSR  13
      DCL STRIND ENTRY(FIXED BIN(31,0),FIXED BIN(31,0))                    ADVSR  14
                 RETURNS(FIXED BIN(31,0));                                 ADVSR  15
      DCL NUCELL ENTRY RETURNS(FIXED BIN(31,0));                           ADVSR  16
      DCL RCELL  ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));          ADVSR  17
      DCL LCNTR  ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));          ADVSR  18
      DCL CAL FIXED BIN(31,0) STATIC;                                      ADVSR  19
      R=CONT(L); CAND=CONT(LNKL(R));                                       ADVSR  20
      IF ID(CAND)=1 THEN GOTO L6;                                          ADVSR  21
L1:   LCP=LNKR(CAND); CALL SETDIR(-1,LCP,-1,R);                            ADVSR  22
      CAND=CONT(LCP);                                                      ADVSR  23
      IF ID(CAND)=2 THEN GOTO L4;                                          ADVSR  24
      IF ID(CAND)=J | ID(CAND)=K THEN GOTO L8;                             ADVSR  25
      IF ID(CAND)¬=1 THEN GOTO L1;                                         ADVSR  26
L6:   M=NUCELL; CAL= STRIND(R,M); CAL= STRIND(CONT(L+1),M+1);              ADVSR  27
      CALL SETIND(-1,LNKR(CONT(LNKL(R)+1)),LCNTR(L)+1,L+1);                ADVSR  28
      CALL SETDIR(-1,-1,M,R);                                              ADVSR  29
      CAND=CONT(INHALT(LNKL(R)+1));                                        ADVSR  30
      GOTO L1;                                                             ADVSR  31
L4:   IF LCNTR(L)¬=0 THEN GOTO L9;                                         ADVSR  32
      ADVSA=-1; GOTO L12;                                                  ADVSR  33
L9:   LK=LNKR(R); R=CONT(LK); CAL= STRIND(CONT(LK+1),L+1);                 ADVSR  34
      CAND=CONT(LNKL(R)); CALL RCELL(LK);                                  ADVSR  35
      GOTO L1;                                                             ADVSR  36
L8:   ADVSA=0;                                                             ADVSR  37
```

```
   L12: CAL= STRIND(R,L);                                                    ADVSR   38
        RETURN(ADVSA);                                                       ADVSR   39
        END ADVSR;                                                           ADVSR   40


ADVSWL: PROC(LR,A)          FIXED BIN(31,0);                                 ADVSWL   0
        DCL (LR,A)          FIXED BIN(31,0);                                 ADVSWL   1
        DCL REED   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));          ADVSWL   2
        DCL ADVSL ENTRY(FIXED BIN(31,0),FIXED BIN(31,0),                     ADVSWL   3
        FIXED BIN(31,0))RETURNS(FIXED BIN(31,0));                            ADVSWL   4
        A=ADVSL(LR,1,0);                                                     ADVSWL   5
        IF A=0 THEN RETURN(REED(LR));                                        ADVSWL   6
        RETURN (A);                                                          ADVSWL   7
        END ADVSWL;                                                          ADVSWL   8


ADVSWR: PROC(LR,A)          FIXED BIN(31,0);                                 ADVSWR   0
        DCL (LR,A)          FIXED BIN(31,0);                                 ADVSWR   1
        DCL REED   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));          ADVSWR   2
        DCL ADVSR ENTRY(FIXED BIN(31,0),FIXED BIN(31,0),                     ADVSWR   3
        FIXED BIN(31,0))RETURNS(FIXED BIN(31,0));                            ADVSWR   4
        A=ADVSR(LR,1,0);                                                     ADVSWR   5
        IF A=0 THEN RETURN(REED(LR));                                        ADVSWR   6
        RETURN (A);                                                          ADVSWR   7
        END ADVSWR;                                                          ADVSWR   8


BOT:                                                                         BOT      0
B:      PROC(P)             FIXED BIN(31,0);                                 BOT      1
        DCL (P)             FIXED BIN(31,0);                                 BOT      2
        DCL CONT   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));          BOT      3
```

```
        DCL LNKL   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));          BOT      4
        DCL LOCT   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));          BOT      5
        RETURN(CONT(LNKL(CONT(LOCT(P)))+1));                                 BOT      6
        END BOT;                                                             BOT      7


DELETE: PROC(K)              FIXED BIN(31,0);                                DELETE   0
        DCL (K)              FIXED BIN(31,0);                                DELETE   1
        DCL CONT   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));          DELETE   2
        DCL ID     ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));          DELETE   3
        DCL LNKL   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));          DELETE   4
        DCL LNKR   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));          DELETE   5
        DCL SETIND ENTRY(FIXED BIN(31,0),FIXED BIN(31,0),                    DELETE   6
                    FIXED BIN(31,0),FIXED BIN(31,0))                         DELETE   7
                   RETURNS(FIXED BIN(31,0));                                 DELETE   8
        DCL RCELL  ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));          DELETE   9
        DCL (LL,LR)          FIXED BIN(31,0) STATIC;                         DELETE  10
        IF ID(CONT(K))=2 THEN DO;                                            DELETE  11
                  PUT LIST (' AN ATTEMPT HAS BEEN MADE TO DELETE A ',        DELETE  12
                  'HEADER _ ZERO HAS BEEN DELIVERED AND THE ',               DELETE  13
                  'PROGRAM CONTINUED');                                      DELETE  14
                  RETURN(0);                                                 DELETE  15
                  END;                                                       DELETE  16
        LL=LNKL(CONT(K));                                                    DELETE  17
        LR=LNKR(CONT(K));                                                    DELETE  18
        CALL RCELL(K);                                                       DELETE  19
        CALL SETIND(-1,-1,LR,LL);                                            DELETE  20
        CALL SETIND(-1,LL,-1,LR);                                            DELETE  21
        RETURN(CONT(K+1));                                                   DELETE  22
        END DELETE;                                                          DELETE  23


INITAS: PROC(M,N);                                                           INITAS   0
```

```
        DCL SETDIR ENTRY(FIXED BIN(31,0),FIXED BIN(31,0),                    INITAS   1
                         FIXED BIN(31,0),FIXED BIN(31,0))                    INITAS   2
                   RETURNS(FIXED BIN(31,0));                                 INITAS   3
        DCL MADOV  ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));          INITAS   4
        DCL(M(N),N)FIXED BIN(31,0);                                          INITAS   5
        DCL(I,J,K) FIXED BIN(31,0);                                          INITAS   6
        DCL AVSL FIXED BIN(31,0) STATIC EXT;                                 INITAS   7
        DCL RELOC FIXED BIN(31,0) STATIC EXT;                                INITAS   8
        DCL LAVSIZ FIXED BIN(31,0) STATIC EXT;                               INITAS   9
         LAVSIZ=N;                                                           INITAS  10
        RELOC=MADOV(M(1))-4;                                                 INITAS  11
        DO I=1 TO N; M(I)=0; END;                                            INITAS  12
        K=N-2;                                                               INITAS  13
        DO I=1 TO K BY 2;                                                    INITAS  14
        CALL SETDIR(-1,-1,I+2,M(I));                                         INITAS  15
        END;                                                                 INITAS  16
        CALL SETDIR(0,N-1,1,AVSL);                                           INITAS  17
        END INITAS;                                                          INITAS  18


INITRD: PROC(K)   FIXED BIN(31,0);                                           INITRD   0
        DCL (K)   FIXED BIN(31,0);                                           INITRD   1
        DCL CONT   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));          INITRD   2
        DCL LNKL   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));          INITRD   3
        DCL SETIND ENTRY(FIXED BIN(31,0),FIXED BIN(31,0),                    INITRD   4
                         FIXED BIN(31,0),FIXED BIN(31,0))                    INITRD   5
                   RETURNS(FIXED BIN(31,0));                                 INITRD   6
        CALL SETIND(-1,LNKL(CONT(K+1)),-1,K);                                INITRD   7
        RETURN(K);                                                           INITRD   8
        END INITRD;                                                          INITRD   9


INLSTL:                                                                      INLSTL   0
```

```
INL:    PROC(M,N)              FIXED BIN(31,0);                                  INLSTL   1
        DCL (M,N)              FIXED BIN(31,0);                                  INLSTL   2
        DCL (L,ITOP,IBOT,IPRE) FIXED BIN(31,0) STATIC;                           INLSTL   3
        DCL CONT   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));              INLSTL   4
        DCL LNKL   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));              INLSTL   5
        DCL LNKR   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));              INLSTL   6
        DCL SETIND ENTRY(FIXED BIN(31,0),FIXED BIN(31,0),                        INLSTL   7
                         FIXED BIN(31,0),FIXED BIN(31,0))                        INLSTL   8
                   RETURNS(FIXED BIN(31,0));                                     INLSTL   9
        DCL LOCT   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));              INLSTL  10
        L=LOCT(M);                                                               INLSTL  11
        ITOP=LNKR(CONT(L));                                                      INLSTL  12
        IBOT=LNKL(CONT(L));                                                      INLSTL  13
        CALL SETIND(-1,L,L,L);                                                   INLSTL  14
        IPRE=LNKL(CONT(N));                                                      INLSTL  15
        CALL SETIND(-1,IBOT,-1,N);                                               INLSTL  16
        CALL SETIND(-1,-1,ITOP,IPRE);                                            INLSTL  17
        CALL SETIND(-1,IPRE,-1,ITOP);                                            INLSTL  18
        CALL SETIND(-1,-1,N,IBOT);                                               INLSTL  19
        RETURN(L);                                                               INLSTL  20
        END INLSTL;                                                              INLSTL  21


INLSTR:                                                                          INLSTR   0
INR:    PROC(M,N)              FIXED BIN(31,0);                                  INLSTR   1
        DCL (M,N)              FIXED BIN(31,0);                                  INLSTR   2
        DCL (L,ITOP,IBOT,ISLO) FIXED BIN(31,0) STATIC;                           INLSTR   3
        DCL CONT   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));              INLSTR   4
        DCL LNKL   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));              INLSTR   5
        DCL LNKR   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));              INLSTR   6
        DCL SETIND ENTRY(FIXED BIN(31,0),FIXED BIN(31,0),                        INLSTR   7
                         FIXED BIN(31,0),FIXED BIN(31,0))                        INLSTR   8
                   RETURNS(FIXED BIN(31,0));                                     INLSTR   9
```

```
        DCL LOCT   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));          INLSTR  10
        L=LOCT(M);                                                           INLSTR  11
        ITOP=LNKR(CONT(L));                                                  INLSTR  12
        IBOT=LNKL(CONT(L));                                                  INLSTR  13
        CALL SETIND(-1,L,L,L);                                               INLSTR  14
        ISUC=LNKR(CONT(N));                                                  INLSTR  15
        CALL SETIND(-1,-1,ITOP,N);                                           INLSTR  16
        CALL SETIND(-1,IBOT,-1,ISUC);                                        INLSTR  17
        CALL SETIND(-1,N,-1,ITOP);                                           INLSTR  18
        CALL SETIND(-1,-1,ISUC,IBOT);                                        INLSTR  19
        RETURN(L);                                                           INLSTR  20
        END INLSTR;                                                          INLSTR  21


IRALST: PROC(P)   FIXED BIN(31,0);                                           IRALST   0
        DCL CONT   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));          IRALST   1
        DCL LNKL   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));          IRALST   2
        DCL RCELL  ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));          IRALST   3
        DCL SETIND ENTRY(FIXED BIN(31,0),FIXED BIN(31,0),                    IRALST   4
                         FIXED BIN(31,0),FIXED BIN(31,0))                    IRALST   5
                   RETURNS(FIXED BIN(31,0));                                 IRALST   6
        DCL LCNTR  ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));          IRALST   7
        DCL LOCT   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));          IRALST   8
        DCL MTLIST ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));          IRALST   9
        DCL (L,N,IRALSA) FIXED BIN(31,0) STATIC;                             IRALST  10
        DCL P FIXED BIN(31,0);                                               IRALST  11
        DCL CAL FIXED BIN(31,0) STATIC;                                      IRALST  12
                   /* THE LIST P IS ERASED. IF THIS LIST IS THE             IRALST  13
                   SUBLIST OF ANOTHER LIST THE REFERENCE COUNTER             IRALST  14
                   IS COUNTED DOWNAND THE LIST IS NOT ERASED.                IRALST  15
                   THE VALUE OF THE FUNCTION IS THE NUMBER OF LISTS          IRALST  16
                   OF WHICH P IS A SUBLIST. IFTHE VALUE IS ZERO              IRALST  17
                   THE LIST HAS BEEN ERASED */                               IRALST  18
```

```
          L=LUCT(P);                                                    IRALST  19
          CALL SETIND(-1,-1,LCNTR(L)-1,L+1);                            IRALST  20
          IRALSA=LCNTR(L);                                              IRALST  21
          IF IRALSA ¬=0 THEN RETURN(IRALSA);                            IRALST  22
          CAL= MTLIST(P);                                               IRALST  23
          N=LNKL(CONT(L+1));                                            IRALST  24
          IF N ¬=0 THEN DO;                                             IRALST  25
                    CALL SETIND(1,-1,-1,L);                             IRALST  26
                    CALL SETIND(-1,N,N,L+1); END;                       IRALST  27
          CALL RCELL(L);                                                IRALST  28
          RETURN(IRALSA);                                               IRALST  29
          END IRALST;                                                   IRALST  30


IRARDR:   PROC(K)    FIXED BIN(31,0);                                   IRARDR   0
          DCL (K)    FIXED BIN(31,0);                                   IRARDR   1
          DCL (M,N)            FIXED BIN(31,0) STATIC;                 IRARDR   2
          DCL CONT   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));   IRARDR   3
          DCL LNKR   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));   IRARDR   4
          DCL LCNTR  ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));   IRARDR   5
          DCL RCELL  ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));   IRARDR   6
          M=K;                                                          IRARDR   7
    L3:   N=LNKR(CONT(M)); CALL RCELL(M);                               IRARDR   8
          IF N=0 THEN RETURN(LCNTR(K));                                 IRARDR   9
          M=N; GOTO L3;                                                 IRARDR  10
          END IRARDR;                                                   IRARDR  11


LAVSOUT:  PROC(LAVS,M);                                                 LAVSOUT  0
          DCL AVSL FIXED BIN(31,0) STATIC EXT;                          LAVSOUT  1
          DCL(LAVS(M),M)FIXED BIN(31,0);                                LAVSOUT  2
          DCL ID     ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));   LAVSOUT  3
```

```
        DCL LNKR ENTRY(FIXED BIN(31,0))RETURNS(FIXED BIN(31,0));                LAVSOUT  4
        DCL LNKL ENTRY(FIXED BIN(31,0))RETURNS(FIXED BIN(31,0));                LAVSOUT  5
        DCL MADDV ENTRY(FIXED BIN(31,0))RETURNS(FIXED BIN(31,0));               LAVSOUT  6
        DCL I FIXED BIN(31,0);                                                  LAVSOUT  7
        DCL RELOC FIXED BIN(31,0) STATIC EXT;                                   LAVSOUT  8
        PUT SKIP(4);                                                            LAVSOUT  9
        PUT EDIT('AVSL')(A);                                                    LAVSOUT 10
        PUT EDIT(ID(AVSL),LNKL(AVSL),LNKR(AVSL))(F(10));                        LAVSOUT 11
        PUT SKIP(4);                                                            LAVSOUT 12
        PUT EDIT('    ADDR       ID     LNKL     LNKR                '||       LAVSOUT 13
                 '    ID     LNKL     LNKR          DATUM') (A);               LAVSOUT 14
        PUT SKIP(4);                                                            LAVSOUT 15
        PUT EDIT((I,ID(LAVS(I)),LNKL(LAVS(I)),                                  LAVSOUT 16
                  LNKR(LAVS(I)),ID(LAVS(I+1)),LNKL(LAVS(I+1)),                  LAVSOUT 17
                  LNKR(LAVS(I+1)),UNSPEC(LAVS(I+1))                             LAVSOUT 18
                  DO I=1 TO M BY 2))                                            LAVSOUT 19
                  (4 F(8),X(10),3 F(8),X(5),B(32),SKIP);                        LAVSOUT 20
        END LAVSOUT;                                                            LAVSOUT 21


LCNTR:  PROC(K)   FIXED BIN(31,0);                                              LCNTR    0
        DCL CONT   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));             LCNTR    1
        DCL LNKR   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));             LCNTR    2
        DCL K FIXED BIN(31,0);                                                  LCNTR    3
        RETURN(LNKR(CONT(K+1)));                                                LCNTR    4
        END LCNTR;                                                              LCNTR    5


LIST:   PROC(K)   FIXED BIN(31,0);                                              LIST     0
        /* CREATES AN EMPTY LIST AND LEAVES ITS NAME AS ITS VALUE               LIST     1
           AND IN THE CELL K WHICH THEREBY BECOMES AN ALIAS FOR THE LIS         LIST     2
        */                                                                      LIST     3
```

```
        DCL NUCELL ENTRY RETURNS(FIXED BIN(31,0));                          LIST    4
        DCL SETDIR ENTRY(FIXED BIN(31,0),FIXED BIN(31,0),                   LIST    5
                         FIXED BIN(31,0),FIXED BIN(31,0))                   LIST    6
                   RETURNS(FIXED BIN(31,0));                                LIST    7
        DCL SETIND ENTRY(FIXED BIN(31,0),FIXED BIN(31,0),                   LIST    8
                         FIXED BIN(31,0),FIXED BIN(31,0))                   LIST    9
                   RETURNS(FIXED BIN(31,0));                                LIST   10
        DCL LISA FIXED BIN(31,0) STATIC;                                    LIST   11
        DCL K FIXED BIN(31,0);                                              LIST   12
        LISA=NUCELL;                                                        LIST   13
        CALL SETDIR(0,LISA,LISA,LISA);                                      LIST   14
        CALL SETIND(2,LISA,LISA,LISA);                                      LIST   15
        IF K¬=9 THEN DO; CALL SETIND(-1,-1,1,LISA+1); K=LISA; END;          LIST   16
        RETURN(LISA);                                                       LIST   17
        END LIST;                                                           LIST   18


LISTMT: PROC(P)   FIXED BIN(31,0);                                          LISTMT  0
        DCL CONT   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));         LISTMT  1
        DCL LNKR   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));         LISTMT  2
        DCL LOCT   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));         LISTMT  3
        DCL P FIXED BIN(31,0);                                              LISTMT  4
        DCL L FIXED BIN(31,0) STATIC;                                       LISTMT  5
        L=CONT(LOCT(P));                                                    LISTMT  6
        IF L=CONT(LNKR(L)) THEN RETURN(0);                                  LISTMT  7
        RETURN(-1);                                                         LISTMT  8
        END LISTMT;                                                         LISTMT  9


LOCT:   PROC(K)   FIXED BIN(31,0);                                          LOCT    0
        DCL CONT   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));         LOCT    1
        DCL ID     ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));         LOCT    2
```

```
        DCL LNKL   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));          LOCT     3
        DCL LNKR   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));          LOCT     4
        DCL NAMTST ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));          LOCT     5
        DCL K FIXED BIN(31,0);                                               LOCT     6
        IF NAMTST(K)=0 THEN RETURN(K);                                       LOCT     7
        PUT SKIP(2) EDIT ('LIST WAS REQUIRED AS AN OPERAND BUT WAS',         LOCT     8
                  ' NOT FOUND - PROGRAM REGRETFULLY TERMINATED')             LOCT     9
                 (A);                                                        LOCT    10
        PUT SKIP(2) LIST('LNKR=',LNKR(K),'LNKL=',LNKL(K));                   LOCT    11
        SIGNAL CONDITION(ALPHA);                                             LOCT    12
        END LOCT;                                                            LOCT    13


LOFRDR: PROC(K)   FIXED BIN(31,0);                                           LOFRDR   0
        DCL (K)   FIXED BIN(31,0);                                           LOFRDR   1
        DCL L     FIXED BIN(31,0) STATIC;                                    LOFRDR   2
        DCL CONT   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));          LOFRDR   3
        DCL LNKL   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));          LOFRDR   4
        DCL SETDIR ENTRY(FIXED BIN(31,0),FIXED BIN(31,0),                    LOFRDR   5
                         FIXED BIN(31,0),FIXED BIN(31,0))                    LOFRDR   6
                   RETURNS(FIXED BIN(31,0));                                 LOFRDR   7
      L=LNKL(CONT(K+1));                                                     LOFRDR   8
        CALL SETDIR(0,L,L,L);                                                LOFRDR   9
        RETURN(L);                                                           LOFRDR  10
        END LOFRDR;                                                          LOFRDR  11


LPNTR:  PROC(K)              FIXED BIN(31,0);                                LPNTR    0
        DCL(K)               FIXED BIN(31,0);                                LPNTR    1
        DCL CONT   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));          LPNTR    2
        DCL LNKL   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));          LPNTR    3
        RETURN(LNKL(CONT(K)));                                               LPNTR    4
```

```
        END LPNTR;                                                          LPNTR    5


LPURGE: PROC(LST)          FIXED BIN(31,0);                                 LPURGE   0
        DCL (LST)          FIXED BIN(31,0);                                 LPURGE   1
        DCL (X,LP,K,J,L)   FIXED BIN(31,0) STATIC;                          LPURGE   2
         DCL CONT  ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));         LPURGE   3
         DCL LNKR  ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));         LPURGE   4
         DCL SETIND ENTRY(FIXED BIN(31,0),FIXED BIN(31,0),                  LPURGE   5
                    FIXED BIN(31,0),FIXED BIN(31,0))                        LPURGE   6
                    RETURNS(FIXED BIN(31,0));                               LPURGE   7
         DCL LCNTR  ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));        LPURGE   8
         DCL NAMTST ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));        LPURGE   9
         DCL LRDROV ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));        LPURGE  10
         DCL IRARDR ENTRY(FIXED BIN(31,0))RETURNS(FIXED BIN(31,0));         LPURGE  11
         DCL DELETE ENTRY(FIXED BIN(31,0))RETURNS(FIXED BIN(31,0));         LPURGE  12
         DCL LPNTR  ENTRY(FIXED BIN(31,0))RETURNS(FIXED BIN(31,0));         LPURGE  13
         DCL LSTPRO ENTRY(FIXED BIN(31,0),FIXED BIN(31,0))                  LPURGE  14
                    RETURNS(FIXED BIN(31,0));                               LPURGE  15
         DCL ADVSWR ENTRY(FIXED BIN(31,0),FIXED BIN(31,0))                  LPURGE  16
                    RETURNS(FIXED BIN(31,0));                               LPURGE  17
         DCL ADVLWR ENTRY(FIXED BIN(31,0),FIXED BIN(31,0))                  LPURGE  18
                    RETURNS(FIXED BIN(31,0));                               LPURGE  19
        DCL CAL FIXED BIN(31,0) STATIC;                                     LPURGE  20
        LP=0;                                                               LPURGE  21
    L9: K=LRDROV(LST);                                                      LPURGE  22
    L3: X=ADVSWR(K,J);                                                      LPURGE  23
    L6: IF J¬=0 THEN GOTO L2;                                               LPURGE  24
        IF NAMTST(X)¬=0 | LSTPRO(X,K)¬=0 THEN GOTO L3;                      LPURGE  25
        L=LPNTR(K);                                                         LPURGE  26
        CALL SETIND(-1,-1,LCNTR(X)-1,X+1);                                  LPURGE  27
        X=ADVLWR(K,L);                                                      LPURGE  28
        CAL= DELETE(L);                                                     LPURGE  29
```

```
        LP=LP+1;                                                        LPURGE  30
        GOTO L6;                                                        LPURGE  31
    L2: IF LNKR(CONT(K))¬=0 THEN DO;                                    LPURGE  32
                CAL= IRARDR(K);                                         LPURGE  33
                GOTO L9;                                                LPURGE  34
                END;                                                    LPURGE  35
        CAL= IRARDR(K);                                                 LPURGE  36
        RETURN (LP);                                                    LPURGE  37
        END LPURGE;                                                     LPURGE  38


LRDRCP: PROC(K)   FIXED BIN(31,0);                                      LRDRCP   0
        DCL (K)   FIXED BIN(31,0);                                      LRDRCP   1
       DCL (LR,NEWR,NOW)   FIXED BIN(31,0) STATIC;                      LRDRCP   2
        DCL CONT   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));     LRDRCP   3
        DCL LNKR   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));     LRDRCP   4
        DCL SETIND ENTRY(FIXED BIN(31,0),FIXED BIN(31,0),               LRDRCP   5
                         FIXED BIN(31,0),FIXED BIN(31,0))               LRDRCP   6
                   RETURNS(FIXED BIN(31,0));                            LRDRCP   7
        DCL STRIND ENTRY(FIXED BIN(31,0),FIXED BIN(31,0))               LRDRCP   8
                   RETURNS(FIXED BIN(31,0));                            LRDRCP   9
        DCL NUCELL ENTRY RETURNS(FIXED BIN(31,0));                      LRDRCP  10
        DCL NEW FIXED BIN(31,0);                                        LRDRCP  11
        DCL CAL FIXED BIN(31,0) STATIC;                                 LRDRCP  12
        LR=NUCELL;                                                      LRDRCP  13
        NEWR=LR;                                                        LRDRCP  14
        NOW=K;                                                          LRDRCP  15
    L3: CAL= STRIND(CONT(NOW),NEWR);                                    LRDRCP  16
        CAL= STRIND(CONT(NOW+1),NEWR+1);                                LRDRCP  17
        NOW=LNKR(CONT(NOW));                                            LRDRCP  18
        IF NOW=0 THEN RETURN(LR);                                       LRDRCP  19
        NEW=NUCELL;                                                     LRDRCP  20
        CALL SETIND(-1,-1,NEW,NEWR);                                    LRDRCP  21
```

```
      NEWR=NEW; GOTO L3;                                                LRDRCP  22
      END LRDRCP;                                                       LRDRCP  23


LRDROV: PROC(P)   FIXED BIN(31,0);                                      LRDROV   0
      DCL (P)   FIXED BIN(31,0);                                        LRDROV   1
       DCL SETIND ENTRY(FIXED BIN(31,0),FIXED BIN(31,0),                LRDROV   2
                       FIXED BIN(31,0),FIXED BIN(31,0))                 LRDROV   3
                  RETURNS(FIXED BIN(31,0));                             LRDROV   4
       DCL LOCT   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));      LRDROV   5
       DCL NUCELL ENTRY RETURNS(FIXED BIN(31,0));                       LRDROV   6
      DCL LRDROA FIXED BIN(31,0) STATIC;                                LRDROV   7
       LRDROA=NUCELL;                                                   LRDROV   8
       CALL SETIND(3,LOCT(P),0,LRDROA);                                 LRDROV   9
       CALL SETIND(0,P,0,LRDROA+1);                                     LRDROV  10
       RETURN(LRDROA);                                                  LRDROV  11
       END LRDROV;                                                      LRDROV  12


LSSCPY: PROC(LA)          FIXED BIN(31,0) RECURSIVE;                    LSSCPY   0
      DCL (LA)            FIXED BIN(31,0);                              LSSCPY   1
       DCL (LB,LR,LI,LRA,LRB,IDA,IDB) FIXED BIN(31,0);                  LSSCPY   2
       DCL CONT   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));      LSSCPY   3
       DCL LIST   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));      LSSCPY   4
       DCL CAL FIXED BIN(31,0)STATIC;                                   LSSCPY   5
       DCL ID     ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));      LSSCPY   6
       DCL LNKR ENTRY(FIXED BIN(31,0))RETURNS(FIXED BIN(31,0));         LSSCPY   7
       DCL LSSCPY ENTRY(FIXED BIN(31,0))RETURNS(FIXED BIN(31,0));       LSSCPY   8
       DCL NEWBOT ENTRY(FIXED BIN(31,0),FIXED BIN(31,0))                LSSCPY   9
                  RETURNS(FIXED BIN(31,0));                             LSSCPY  10
       DCL LOCT   ENTRY(FIXED BIN(31,0))RETURNS(FIXED BIN(31,0));       LSSCPY  11
       LB=LIST(LB); LR=LNKR(CONT(LA));                                  LSSCPY  12
```

```
     TOP: LI=ID(CONT(LR));                                                    LSSCPY  13
          IF LI=0 THEN DO; CAL= NEWBOT(CONT(LR+1),LB);                        LSSCPY  14
                           LR=LNKR(CONT(LR));                                 LSSCPY  15
                           GOTO TOP;                                          LSSCPY  16
                       END;                                                   LSSCPY  17
          IF LI=1 THEN DO; CAL=NEWBOT(LSSCPY(LOCT(CONT(LR+1))),LB);           LSSCPY  18
                           LR=LNKR(CONT(LR));                                 LSSCPY  19
                           GOTO TOP;                                          LSSCPY  20
                       END;                                                   LSSCPY  21
        IF LI=2 THEN RETURN(LB);                                              LSSCPY  22
        PUT LIST('INVALID ID FIELD ENCOUNTERE IN LSSCPY'); STOP;              LSSCPY  23
         END LSSCPY;                                                          LSSCPY  24


LSTEQL: PROC(LA,LB)          FIXED BIN(31,0) RECURSIVE;                       LSTEQL   0
        DCL (LA,LB)          FIXED BIN(31,0);                                 LSTEQL   1
         DCL (IDA,IDB,LRA,LRB)FIXED BIN(31,0);                               LSTEQL   2
         DCL CONT   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));         LSTEQL   3
         DCL ID     ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));         LSTEQL   4
         DCL LNKR ENTRY(FIXED BIN(31,0))RETURNS(FIXED BIN(31,0));            LSTEQL   5
         DCL LSTEQL ENTRY(FIXED BIN(31,0),FIXED BIN(31,0))                   LSTEQL   6
                    RETURNS(FIXED BIN(31,0));                                LSTEQL   7
        LRA=LNKR(CONT(LA)); LRB= LNKR(CONT(LB));                             LSTEQL   8
     TOP: IDA=ID(CONT(LRA)); IDB=ID(CONT(LRB));                              LSTEQL   9
        IF IDA¬=IDB THEN RETURN(-1);                                         LSTEQL  10
        IF IDA=0 THEN IF CONT(LRA+1)=CONT(LRB+1)                             LSTEQL  11
                      THEN DO LRA=LNKR(CONT(LRA));                           LSTEQL  12
                              LRB=LNKR(CONT(LRB));                           LSTEQL  13
                              GOTO TOP;                                      LSTEQL  14
                          END;                                               LSTEQL  15
                     ELSE RETURN(-1);                                        LSTEQL  16
         IF IDA=1 THEN IF LSTEQL(CONT(LRA+1),CONT(LRB+1))=0                  LSTEQL  17
                      THEN DO LRA=LNKR(CONT(LRA));                           LSTEQL  18
```

```
                        LRB=LNKR(CONT(LRB));                                    LSTEQL  19
                        GOTO TOP;                                               LSTEQL  20
                      END;                                                      LSTEQL  21
                    ELSE RETURN(-1);                                            LSTEQL  22
      IF IDA=2 THEN RETURN(0);                                                  LSTEQL  23
      PUT LIST('INVALID ID FIELD ENCOUNTERED IN LSTEQL');                       LSTEQL  24
       STOP;                                                                    LSTEQL  25
       END LSTEQL;                                                              LSTEQL  26


LSTPRO: PROC(L,K)           FIXED BIN(31,0);                                    LSTPRO   0
       DCL(L,K)             FIXED BIN(31,0);                                    LSTPRO   1
      DCL (NEXT)            FIXED BIN(31,0) STATIC;                             LSTPRO   2
       DCL CONT   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));              LSTPRO   3
       DCL LNKL   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));              LSTPRO   4
       DCL LNKR   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));              LSTPRO   5
       NEXT=K;                                                                  LSTPRO   6
    L3:  IF LNKL(CONT(NEXT+1))=LNKR(L) THEN RETURN (0);                         LSTPRO   7
       NEXT=LNKR(CONT(NEXT));                                                   LSTPRO   8
       IF NEXT¬=0 THEN GOTO L3;                                                 LSTPRO   9
      RETURN(-1);                                                               LSTPRO  10
       END LSTPRO;                                                              LSTPRO  11


LVLRV1:  PROC(K)   FIXED BIN(31,0);                                             LVLRV1   0
       DCL (K)   FIXED BIN(31,0);                                               LVLRV1   1
      DCL (LVLRVA,L)      FIXED BIN(31,0) STATIC;                               LVLRV1   2
       DCL CONT   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));              LVLRV1   3
       DCL LNKR   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));              LVLRV1   4
       DCL LCNTR  ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));              LVLRV1   5
       DCL RCELL  ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));              LVLRV1   6
       DCL STRIND ENTRY(FIXED BIN(31,0),FIXED BIN(31,0))                        LVLRV1   7
```

```
                RETURNS(FIXED BIN(31,0));                                  LVLRV1   8
        DCL CAL FIXED BIN(31,0) STATIC;                                    LVLRV1   9
        LVLRVA=K;                                                          LVLRV1  10
        IF LCNTR(K)=0 THEN RETURN(LVLRVA);                                 LVLRV1  11
        L=LNKR(CONT(LVLRVA));                                              LVLRV1  12
        CAL= STRIND(CONT(L),LVLRVA);                                       LVLRV1  13
        CAL= STRIND(CONT(L+1),LVLRVA+1);                                   LVLRV1  14
        CALL RCELL(L);                                                     LVLRV1  15
        RETURN(LVLRVA);                                                    LVLRV1  16
        END LVLRV1;                                                        LVLRV1  17


LVLRVT: PROC(K)   FIXED BIN(31,0);                                         LVLRVT   0
        DCL (K)   FIXED BIN(31,0);                                         LVLRVT   1
       DCL (LVLRVA,L)      FIXED BIN(31,0) STATIC;                         LVLRVT   2
        DCL CONT   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));        LVLRVT   3
        DCL LNKR   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));        LVLRVT   4
        DCL STRIND ENTRY(FIXED BIN(31,0),FIXED BIN(31,0))                  LVLRVT   5
                   RETURNS(FIXED BIN(31,0));                               LVLRVT   6
        DCL LCNTR  ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));        LVLRVT   7
        DCL RCELL  ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));        LVLRVT   8
        DCL CAL FIXED BIN(31,0) STATIC;                                    LVLRVT   9
        LVLRVA=K;                                                          LVLRVT  10
   L1:  IF LCNTR(K)=0 THEN RETURN(LVLRVA) ;                                LVLRVT  11
       L=LNKR(CONT(LVLRVA));                                               LVLRVT  12
        CAL= STRIND(CONT(L),LVLRVA);                                       LVLRVT  13
        CAL= STRIND(CONT(L+1),LVLRVA+1);                                   LVLRVT  14
        CALL RCELL(L); GOTO L1;                                            LVLRVT  15
        END LVLRVT;                                                        LVLRVT  16


MADLFT:                                                                    MADLFT   0
```

```
ML:     PROC(K)  FIXED BIN(31,0);                                          MADLFT   1
        DCL (K)  FIXED BIN(31,0);                                          MADLFT   2
       DCL M    FIXED BIN(31,0) STATIC;                                    MADLFT   3
        DCL CONT   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));        MADLFT   4
        DCL ID     ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));        MADLFT   5
        DCL LNKL   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));        MADLFT   6
        DCL SETDIR ENTRY(FIXED BIN(31,0),FIXED BIN(31,0),                  MADLFT   7
                   FIXED BIN(31,0),FIXED BIN(31,0))                        MADLFT   8
                   RETURNS(FIXED BIN(31,0));                               MADLFT   9
        M=LNKL(CONT(K));                                                   MADLFT  10
        IF  ID(CONT(M))=2 THEN CALL SETDIR(0,M,M,M) ;                      MADLFT  11
        RETURN(M);                                                         MADLFT  12
        END MADLFT;                                                        MADLFT  13


MADNBT: PROC(P,N)          FIXED BIN(31,0);                                MADNBT   0
       DCL (P,N)           FIXED BIN(31,0);                                MADNBT   1
        DCL L      FIXED BIN(31,0) STATIC;                                 MADNBT   2
        DCL CONT   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));        MADNBT   3
        DCL ID     ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));        MADNBT   4
        DCL LNKL   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));        MADNBT   5
        DCL SETDIR ENTRY(FIXED BIN(31,0),FIXED BIN(31,0),                  MADNBT   6
                   FIXED BIN(31,0),FIXED BIN(31,0))                        MADNBT   7
                   RETURNS(FIXED BIN(31,0));                               MADNBT   8
        DCL LOCT   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));        MADNBT   9
       DCL 1     FIXED BIN(31,0) STATIC;                                   MADNBT  10
        L=LOCT(P);                                                         MADNBT  11
        DO I=1 TO N ; L=LNKL(CONT(L)); END;                                MADNBT  12
        IF ID(CONT(L))=2 THEN CALL SETDIR(0,L,L,L);                        MADNBT  13
        RETURN (L);                                                        MADNBT  14
        END MADNBT;                                                        MADNBT  15


MADNTP: PROC(P,N)          FIXED BIN(31,0);                                MADNTP   0
```

```
        DCL (P,N)              FIXED BIN(31,0);                                  MADNTP   1
         DCL L      FIXED BIN(31,0) STATIC;                                     MADNTP   2
         DCL CONT    ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));           MADNTP   3
         DCL ID      ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));           MADNTP   4
         DCL LNKR    ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));           MADNTP   5
         DCL SETDIR ENTRY(FIXED BIN(31,0),FIXED BIN(31,0),                      MADNTP   6
                     FIXED BIN(31,0),FIXED BIN(31,0))                           MADNTP   7
                    RETURNS(FIXED BIN(31,0));                                   MADNTP   8
         DCL LOCT    ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));           MADNTP   9
        DCL I       FIXED BIN(31,0) STATIC;                                     MADNTP  10
         L=LOCT(P);                                                             MADNTP  11
        DO I=1 TO N; L=LNKR(CONT(L)); END;                                      MADNTP  12
         IF ID(CONT(L))=2 THEN CALL SETDIR(0,L,L,L);                            MADNTP  13
         RETURN (L);                                                            MADNTP  14
         END MADNTP;                                                            MADNTP  15


MADRGT:                                                                         MADRGT   0
MR:      PROC(K)    FIXED BIN(31,0);                                           MADRGT   1
         DCL (K)    FIXED BIN(31,0);                                           MADRGT   2
        DCL M       FIXED BIN(31,0) STATIC;                                     MADRGT   3
         DCL CONT    ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));           MADRGT   4
         DCL ID      ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));           MADRGT   5
         DCL LNKR    ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));           MADRGT   6
         DCL SETDIR ENTRY(FIXED BIN(31,0),FIXED BIN(31,0),                      MADRGT   7
                     FIXED BIN(31,0),FIXED BIN(31,0))                           MADRGT   8
                    RETURNS(FIXED BIN(31,0));                                   MADRGT   9
         M=LNKR(CONT(K));                                                       MADRGT  10
         IF ID(CONT(M))=2 THEN CALL SETDIR(0,M,M,M);                            MADRGT  11
         RETURN (M);                                                            MADRGT  12
         END MADRGT;                                                            MADRGT  13


MRKLSS:  PROC(M,LST)          FIXED BIN(31,0);                                  MRKLSS   0
```

```
        DCL (M,LST)          FIXED BIN(31,0);                                  MRKLSS   1
        DCL LNKR   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));            MRKLSS   2
        DCL SETIND ENTRY(FIXED BIN(31,0),FIXED BIN(31,0),                      MRKLSS   3
                    FIXED BIN(31,0),FIXED BIN(31,0))                           MRKLSS   4
                   RETURNS(FIXED BIN(31,0));                                   MRKLSS   5
        DCL RCELL  ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));            MRKLSS   6
        DCL LRDRCV ENTRY(FIXED BIN(31,0))RETURNS(FIXED BIN(31,0));             MRKLSS   7
        DCL ADVSNR ENTRY(FIXED BIN(31,0),FIXED BIN(31,0))                      MRKLSS   8
                   RETURNS(FIXED BIN(31,0));                                   MRKLSS   9
        DCL MRKLST ENTRY(FIXED BIN(31,0),FIXED BIN(31,0))                      MRKLSS  10
                   RETURNS(FIXED BIN(31,0));                                   MRKLSS  11
        DCL (K,LR,X) FIXED BIN(31,0);                                          MRKLSS  12
        LR=LRDRCV(MRKLST(M,LST));                                              MRKLSS  13
    L3: X=ADVSNR(LR,K);                                                        MRKLSS  14
        IF K¬=0 THEN DO; CALL RCELL(LR); RETURN (LST); END;                    MRKLSS  15
        CALL SETIND(M,-1,-1,LNKR(X)+1);                                        MRKLSS  16
        GOTO L3;                                                               MRKLSS  17
        END MRKLSS;                                                            MRKLSS  18


MRKLST: PROC(M,LST)          FIXED BIN(31,0);                                  MRKLST   0
        DCL (M,LST)          FIXED BIN(31,0);                                  MRKLST   1
        DCL(LR,X,K)          FIXED BIN(31,0) STATIC;                           MRKLST   2
        DCL SETIND ENTRY(FIXED BIN(31,0),FIXED BIN(31,0),                      MRKLST   3
                    FIXED BIN(31,0),FIXED BIN(31,0))                           MRKLST   4
                   RETURNS(FIXED BIN(31,0));                                   MRKLST   5
        DCL LOCT   ENTRY(FIXED BIN(31,0))RETURNS(FIXED BIN(31,0));             MRKLST   6
        CALL SETIND(M,-1,-1,LOCT(LST)+1);                                      MRKLST   7
        RETURN (LST);                                                          MRKLST   8
        END MRKLST;                                                            MRKLST   9


MTLIST: PROC(P)   FIXED BIN(31,0);                                             MTLIST   0
```

```
                        /* RESTORES LIST P TO LAVS AND LEAVES          MTLIST   1
                           THE NAME O FTHE EMPTY LIST AS ITS           MTLIST   2
                          VALUE */                                     MTLIST   3
      DCL CONT   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));      MTLIST   4
      DCL LNKL   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));      MTLIST   5
      DCL LNKR   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));      MTLIST   6
      DCL SETDIR ENTRY(FIXED BIN(31,0),FIXED BIN(31,0),                MTLIST   7
                       FIXED BIN(31,0),FIXED BIN(31,0))                MTLIST   8
                 RETURNS(FIXED BIN(31,0));                             MTLIST   9
      DCL SETIND ENTRY(FIXED BIN(31,0),FIXED BIN(31,0),                MTLIST  10
                       FIXED BIN(31,0),FIXED BIN(31,0))                MTLIST  11
                 RETURNS(FIXED BIN(31,0));                             MTLIST  12
      DCL LISTMT ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));      MTLIST  13
      DCL LOCT   ENTRY(FIXED BIN(31,0))RETURNS(FIXED BIN(31,0));       MTLIST  14
      DCL(M,LR,LL) FIXED BIN(31,0) STATIC;                             MTLIST  15
      DCL P FIXED BIN(31,0);                                           MTLIST  16
      DCL AVSL FIXED BIN(31,0) STATIC EXT;                             MTLIST  17
       M=LOCT(P);                                                      MTLIST  18
       IF LISTMT(P)=0 THEN RETURN(M);                                  MTLIST  19
       LR =LNKR(CONT(M)); LL=LNKL(CONT(M)); CALL SETIND(-1,M,M,M);     MTLIST  20
       CALL SETIND(-1,-1,LR,LNKL(AVSL));                               MTLIST  21
       CALL SETDIR(-1,LL,-1,AVSL);                                     MTLIST  22
  L5:  CALL SETIND(-1,-1,0,LNKL(AVSL));                                MTLIST  23
      RETURN(M);                                                       MTLIST  24
       END MTLIST;                                                     MTLIST  25



NAMTST:  PROC(K)   FIXED BIN(31,0);                                    NAMTST   0
      DCL CONT   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));      NAMTST   1
      DCL LNKR   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));      NAMTST   2
      DCL LNKL   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));      NAMTST   3
      DCL ID     ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));      NAMTST   4
      DCL K FIXED BIN(31,0);                                           NAMTST   5
```

```
        DCL LAVSIZ FIXED BIN(31,0) STATIC EXT;                                  NAMTST   6
         IF LNKR(K)>LAVSIZ-1 THEN RETURN(-1);                                   NAMTST   7
        IF LNKL(K)¬=LNKR(K) THEN RETURN(-1);                                    NAMTST   8
        IF ID(CONT(K))¬=2 THEN RETURN(-1);                                      NAMTST   9
        IF CONT(LNKR(CONT(LNKL(CONT(K)))))¬=CONT(K) THEN RETURN(-1);            NAMTST  10
        RETURN(0);                                                              NAMTST  11
        END NAMTST;                                                             NAMTST  12


NEWBOT:                                                                         NEWBOT   0
NB:     PROC(P,Q)           FIXED BIN(31,0);                                    NEWBOT   1
        DCL (P,Q)           FIXED BIN(31,0);                                    NEWBOT   2
        DCL LOCT   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));             NEWBOT   3
        DCL NXTLFT ENTRY(FIXED BIN(31,0),FIXED BIN(31,0))                       NEWBOT   4
                   RETURNS(FIXED BIN(31,0));                                    NEWBOT   5
        RETURN(NXTLFT(P,LOCT(Q)));                                              NEWBOT   6
        END NEWBOT;                                                             NEWBOT   7


NEWTOP:                                                                         NEWTOP   0
NT:     PROC(P,Q)           FIXED BIN(31,0);                                    NEWTOP   1
        DCL (P,Q)           FIXED BIN(31,0);                                    NEWTOP   2
        DCL LOCT   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));             NEWTOP   3
        DCL NXTRGT ENTRY(FIXED BIN(31,0),FIXED BIN(31,0))                       NEWTOP   4
                   RETURNS(FIXED BIN(31,0));                                    NEWTOP   5
        RETURN(NXTRGT(P,LOCT(Q)));                                              NEWTOP   6
        END NEWTOP;                                                             NEWTOP   7


NUCELL: PROC      FIXED BIN(31,0);                                              NUCELL   0
        /* TAKES A CELL FROM LAVS AND RETURNS MACHINE ADDRESS */                NUCELL   1
```

```
        DCL CONT   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));          NUCELL   2
        DCL LNKR   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));          NUCELL   3
        DCL ID     ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));          NUCELL   4
        DCL STRIND ENTRY(FIXED BIN(31,0),FIXED BIN(31,0))                    NUCELL   5
                   RETURNS(FIXED BIN(31,0));                                 NUCELL   6
        DCL SETDIR ENTRY(FIXED BIN(31,0),FIXED BIN(31,0),                    NUCELL   7
                         FIXED BIN(31,0),FIXED BIN(31,0))                    NUCELL   8
                   RETURNS(FIXED BIN(31,0));                                 NUCELL   9
        DCL IRALST ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));          NUCELL  10
        DCL       M FIXED BIN(31,0) STATIC;                                  NUCELL  11
        DCL AVSL FIXED BIN(31,0) STATIC EXT;                                 NUCELL  12
        DCL CAL FIXED BIN(31,0) STATIC;                                      NUCELL  13
        M=LNKR(AVSL);                                                        NUCELL  14
        IF M=0 THEN DO;                                                      NUCELL  15
                PUT EDIT (' LIST OF AVAILABLE SPACE',                        NUCELL  16
                  ' EXHAUSTED - PROGRAM TERMINATED') (A);                    NUCELL  17
        SIGNAL CONDITION(ALPHA);                                             NUCELL  18
        STOP;                                                                NUCELL  19
        END;                                                                 NUCELL  20
              ELSE IF ID(CONT(M))=1  THEN CAL= IRALST(CONT(M+1));            NUCELL  21
        CALL SETDIR (-1,-1,LNKR(CONT(M)),AVSL);                              NUCELL  22
        CAL= STRIND(0,M); CAL= STRIND(0,M+1);                                NUCELL  23
        RETURN(M);                                                           NUCELL  24
        END NUCELL;                                                          NUCELL  25



NULSTL: PROC(LNKP,LNKH)     FIXED BIN(31,0);                                 NULSTL   0
        DCL (LNKP,LNKH)     FIXED BIN(31,0);                                 NULSTL   1
        DCL (NULSTA,LTOP,LSUC) FIXED BIN(31,0) STATIC;                       NULSTL   2
        DCL CONT   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));          NULSTL   3
        DCL ID     ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));          NULSTL   4
        DCL LNKR   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));          NULSTL   5
        DCL SETIND ENTRY(FIXED BIN(31,0),FIXED BIN(31,0),                    NULSTL   6
```

```
                  FIXED BIN(31,0),FIXED BIN(31,0))                    NULSTL   7
      RETURNS(FIXED BIN(31,0));                                       NULSTL   8
   DCL LIST   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));        NULSTL   9
    NULSTA=LIST(9);                                                   NULSTL  10
    IF ID (CONT(LNKP))=2 THEN RETURN(NULSTA);                         NULSTL  11
    LTOP=LNKR(CONT(LNKH));                                            NULSTL  12
    LSUC=LNKR(CONT(LNKP));                                            NULSTL  13
    CALL SETIND(-1,-1,LSUC,LNKH);                                     NULSTL  14
    CALL SETIND(-1,LNKH,-1,LSUC);                                     NULSTL  15
    CALL SETIND(2,LNKP,LTOP,NULSTA);                                  NULSTL  16
    CALL SETIND(-1,-1,NULSTA,LNKP);                                   NULSTL  17
   CALL SETIND(-1,NULSTA,-1,LTOP);                                    NULSTL  18
   RETURN(NULSTA);                                                    NULSTL  19
   END NULSTL;                                                        NULSTL  20


NULSTR: PROC(LNKP,LNKH)     FIXED BIN(31,0);                          NULSTR   0
   DCL (LNKP,LNKH)          FIXED BIN(31,0);                          NULSTR   1
   DCL (NULSTA,LBOT,LPRE) FIXED BIN(31,0) STATIC;                     NULSTR   2
   DCL CONT   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));        NULSTR   3
   DCL ID     ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));        NULSTR   4
   DCL LNKL   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));        NULSTR   5
   DCL SETIND ENTRY(FIXED BIN(31,0),FIXED BIN(31,0),                  NULSTR   6
                    FIXED BIN(31,0),FIXED BIN(31,0))                  NULSTR   7
              RETURNS(FIXED BIN(31,0));                               NULSTR   8
   DCL LIST   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));        NULSTR   9
   NULSTA=LIST(9);                                                    NULSTR  10
   IF ID(CONT(LNKP))=2 THEN RETURN(NULSTA);                           NULSTR  11
   LBOT=LNKL(CONT(LNKH));                                             NULSTR  12
   LPRE=LNKL(CONT(LNKP));                                             NULSTR  13
   CALL SETIND(-1,LPRE,-1,LNKH);                                      NULSTR  14
   CALL SETIND(-1,-1,LNKH,LPRE);                                      NULSTR  15
   CALL SETIND(2,LBOT,LNKP,NULSTA);                                   NULSTR  16
```

```
        CALL SETIND(-1,-1,NULSTA,LBOT);                                  NULSTR  17
        CALL SETIND(-1,NULSTA,-1,LNKP);                                  NULSTR  18
        RETURN(NULSTA);                                                  NULSTR  19
        END NULSTR;                                                      NULSTR  20


NXTLFT:                                                                  NXTLFT   0
NXL:    PROC(M,A)           FIXED BIN(31,0);                             NXTLFT   1
        DCL(M,A)            FIXED BIN(31,0);                             NXTLFT   2
        DCL CONT   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));      NXTLFT   3
        DCL LNKL   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));      NXTLFT   4
        DCL SETIND ENTRY(FIXED BIN(31,0),FIXED BIN(31,0),                NXTLFT   5
                         FIXED BIN(31,0),FIXED BIN(31,0))                NXTLFT   6
                   RETURNS(FIXED BIN(31,0));                             NXTLFT   7
        DCL STRIND ENTRY(FIXED BIN(31,0),FIXED BIN(31,0))                NXTLFT   8
                   RETURNS(FIXED BIN(31,0));                             NXTLFT   9
        DCL LCNTR  ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));      NXTLFT  10
        DCL NUCELL ENTRY RETURNS(FIXED BIN(31,0));                       NXTLFT  11
        DCL NAMTST ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));      NXTLFT  12
        DCL(IL,LL)          FIXED BIN(31,0) STATIC;                      NXTLFT  13
        DCL CAL FIXED BIN(31,0) STATIC;                                  NXTLFT  14
        IL= NUCELL;                                                      NXTLFT  15
        LL=LNKL(CONT(A));                                                NXTLFT  16
        CALL SETIND(-1,-1,IL,LL);                                        NXTLFT  17
        CALL SETIND(-1,IL,-1,A);                                         NXTLFT  18
        CALL SETIND(0,LL,A,IL);                                          NXTLFT  19
        IF NAMTST(M)=0 THEN DO;                                          NXTLFT  20
                            CALL SETIND(1,-1,-1,IL);                     NXTLFT  21
                 CALL SETIND(-1,-1,LCNTR(M)+1,M+1); END;                 NXTLFT  22
        CAL= STRIND(M,IL+1);                                             NXTLFT  23
        RETURN(IL);                                                      NXTLFT  24
        END NXTLFT;                                                      NXTLFT  25


NXTRGT:                                                                  NXTRGT   0
```

```
NXR:    PROC(M,A)                FIXED BIN(31,0);                                  NXTRGT   1
        DCL(M,A)                 FIXED BIN(31,0);                                  NXTRGT   2
        DCL CONT   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));                NXTRGT   3
        DCL LNKR   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));                NXTRGT   4
        DCL SETIND ENTRY(FIXED BIN(31,0),FIXED BIN(31,0),                          NXTRGT   5
                         FIXED BIN(31,0),FIXED BIN(31,0))                          NXTRGT   6
                   RETURNS(FIXED BIN(31,0));                                       NXTRGT   7
        DCL STRIND ENTRY(FIXED BIN(31,0),FIXED BIN(31,0))                          NXTRGT   8
                   RETURNS(FIXED BIN(31,0));                                       NXTRGT   9
        DCL LCNTR  ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));                NXTRGT  10
        DCL NUCELL ENTRY RETURNS(FIXED BIN(31,0));                                 NXTRGT  11
        DCL NAMTST ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));                NXTRGT  12
        DCL(IR,LR)               FIXED BIN(31,0) STATIC;                           NXTRGT  13
        DCL CAL FIXED BIN(31,0) STATIC;                                            NXTRGT  14
        IR=NUCELL;                                                                 NXTRGT  15
        LR=LNKR(CONT(A));                                                          NXTRGT  16
        CALL SETIND(-1,IR,-1,LR);                                                  NXTRGT  17
        CALL SETIND(-1,-1,IR,A);                                                   NXTRGT  18
        CALL SETIND(0,A,LR,IR);                                                    NXTRGT  19
        IF NAMTST(M)=0 THEN DO;                                                    NXTRGT  20
             CALL SETIND(1,-1,-1,IR);                                              NXTRGT  21
             CALL SETIND(-1,-1,LCNTR(M)+1,M+1);                                    NXTRGT  22
             END;                                                                  NXTRGT  23
        CAL= STRIND(M,IR+1);                                                       NXTRGT  24
        RETURN(IR);                                                                NXTRGT  25
        END NXTRGT;                                                                NXTRGT  26


POPBDT:                                                                            POPBDT   0
PB:     PROC(P)                  FIXED BIN(31,0);                                  POPBDT   1
        DCL (P)                  FIXED BIN(31,0);                                  POPBDT   2
        DCL CONT   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));                POPBDT   3
        DCL LNKL   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));                POPBDT   4
```

```
        DCL LOCT   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));          POPBOT   5
        DCL DELETE ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));          POPBOT   6
        RETURN(DELETE(LNKL(CONT(LOCT(P)))));                                 POPBOT   7
        END POPBOT;                                                          POPBOT   8


POPTOP:                                                                      POPTOP   0
PT:     PROC(P)                 FIXED BIN(31,0);                             POPTOP   1
        DCL (P)                 FIXED BIN(31,0);                             POPTOP   2
        DCL CONT   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));          POPTOP   3
        DCL LNKR   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));          POPTOP   4
        DCL LOCT   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));          POPTOP   5
        DCL DELETE ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));          POPTOP   6
        RETURN(DELETE(LNKR(CONT(LOCT(P)))));                                 POPTOP   7
        END POPTOP;                                                          POPTOP   8


RCELL:  PROC(CELL)              FIXED BIN(31,0);                             RCELL    0
                                /* RETURNS THE CELL AT ADDRESS 'CELL' TO     RCELL    1
                                   LAVS */                                   RCELL    2
        DCL LNKL   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));          RCELL    3
        DCL SETDIR ENTRY(FIXED BIN(31,0),FIXED BIN(31,0),                    RCELL    4
                         FIXED BIN(31,0),FIXED BIN(31,0))                    RCELL    5
                   RETURNS(FIXED BIN(31,0));                                 RCELL    6
        DCL SETIND ENTRY(FIXED BIN(31,0),FIXED BIN(31,0),                    RCELL    7
                         FIXED BIN(31,0),FIXED BIN(31,0))                    RCELL    8
                   RETURNS(FIXED BIN(31,0));                                 RCELL    9
        DCL CELL FIXED BIN(31,0);                                            RCELL   10
        DCL AVSL FIXED BIN(31,0) STATIC EXT;                                 RCELL   11
         CALL SETIND(-1,-1,CELL,LNKL(AVSL));                                 RCELL   12
         CALL SETDIR (-1,CELL,-1,AVSL);                                      RCELL   13
        CALL SETIND(-1,-1,0,CELL);                                           RCELL   14
```

```
          END RCELL;                                                                    RCELL    15


REED:     PROC(K)              FIXED BIN(31,0);                                         REED      0
          DCL (K)              FIXED BIN(31,0);                                         REED      1
          DCL CONT   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));                   REED      2
          DCL LNKL   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));                   REED      3
          RETURN(CONT(LNKL(CONT(K))+1));                                                REED      4
          END REED;                                                                     REED      5


SEQLL:    PROC(Z,N)            FIXED BIN(31,0);                                         SEQLL     0
          DCL (Z,N)            FIXED BIN(31,0);                                         SEQLL     1
          DCL L      FIXED BIN(31,0) STATIC;                                           SEQLL     2
          DCL CONT   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));                   SEQLL     3
          DCL ID     ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));                   SEQLL     4
          DCL LNKL   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));                   SEQLL     5
          L=LNKL(Z); Z=CONT(L);                                                         SEQLL     6
          N=ID(Z)-1;                                                                    SEQLL     7
          RETURN (CONT(L+1));                                                           SEQLL     8
          END SEQLL;                                                                    SEQLL     9


SEQLR:    PROC(Z,N)            FIXED BIN(31,0);                                         SEQLR     0
          DCL (Z,N)            FIXED BIN(31,0);                                         SEQLR     1
          DCL L      FIXED BIN(31,0) STATIC;                                           SEQLR     2
          DCL CONT   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));                   SEQLR     3
          DCL LNKR   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));                   SEQLR     4
          DCL ID     ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));                   SEQLR     5
          L=LNKR(Z); Z=CONT(L);                                                         SEQLR     6
          N=ID(Z)-1;                                                                    SEQLR     7
```

```
        RETURN (CONT(L+1));                                                    SEQLR     8
        END SEQLR;                                                             SEQLR     9


       SEQRDR:   PROC(LST)          FIXED BIN(31,0);                                     0
        DCL LST   FIXED BIN(31,0);                                                     1
        DCL CONT   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));                    2
        DCL LOCT   ENTRY(FIXED BIN(31,0))RETURNS(FIXED BIN(31,0));                     3
        RETURN (CONT(LOCT(LST)));                                                      4
        END SEQRDR;                                                                    5


SEQSL:  PROC(Z,N)            FIXED BIN(31,0);                                  SEQSL     0
        DCL (Z,N)            FIXED BIN(31,0);                                  SEQSL     1
        DCL L      FIXED BIN(31,0) STATIC;                                     SEQSL     2
        DCL CONT   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));            SEQSL     3
        DCL ID     ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));            SEQSL     4
        DCL LNKL   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));            SEQSL     5
        DCL LNKR   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));            SEQSL     6
        IF ID(Z)¬=1 THEN GOTO L4;                                              SEQSL     7
        L=LNKL(CONT(CONT(LNKL(CONT(LNKR(Z)))+1)));                             SEQSL     8
        GOTO L3;                                                               SEQSL     9
    L4: L=LNKL(Z);                                                             SEQSL    10
    L3: IF ID(CONT(L))=1 THEN GOTO L2;                                         SEQSL    11
        Z=CONT(L);                                                             SEQSL    12
        N=ID(Z)-1;                                                             SEQSL    13
    L2: L=LNKL(CONT(CONT(L+1)));                                               SEQSL    14
        RETURN (CONT(L+1));                                                    SEQSL    15
        END SEQSL;                                                             SEQSL    16


SEQSR:  PROC(Z,N)            FIXED BIN(31,0);                                  SEQSR     0
```

```
        DCL (Z,N)          FIXED BIN(31,0);                                          SEQSR    1
         DCL L      FIXED BIN(31,0) STATIC;                                         SEQSR    2
         DCL CONT   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));               SEQSR    3
         DCL ID     ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));               SEQSR    4
         DCL LNKL   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));               SEQSR    5
         DCL LNKR   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));               SEQSR    6
         IF ID(Z)¬=1 THEN GOTO L4;                                                 SEQSR    7
         L=LNKR(CONT(LNKL(CONT(LNKR(Z)))+1));                                      SEQSR    8
         GOTO L3;                                                                  SEQSR    9
    L4:  L=LNKR(Z);                                                                SEQSR   10
    L3:  IF ID(CONT(L))=1 THEN GOTO L2;                                            SEQSR   11
         Z=CONT(L);                                                                SEQSR   12
         N=ID(Z)-1;                                                                SEQSR   13
         RETURN (CONT(L+1));                                                       SEQSR   14
    L2:  L=LNKR(CONT(CONT(L+1)));                                                  SEQSR   15
         GOTO L3;                                                                  SEQSR   16
         END SEQSR;                                                                SEQSR   17


SUBSBT:  PROC(DAT,LST)          FIXED BIN(31,0);                                   SUBSBT   0
         DCL (DAT,LST)          FIXED BIN(31,0);                                   SUBSBT   1
         DCL CONT   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));               SUBSBT   2
         DCL LNKL   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));               SUBSBT   3
         DCL SUBST ENTRY(FIXED BIN(31,0),FIXED BIN(31,0))                          SUBSBT   4
           RETURNS(FIXED BIN(31,0));                                               SUBSBT   5
         RETURN(SUBST(DAT,LNKL(CONT(LST))));                                       SUBSBT   6
         END SUBSBT;                                                               SUBSBT   7


SUBST:   PROC(D,N)              FIXED BIN(31,0);                                   SUBST    0
         DCL (D,N)              FIXED BIN(31,0);                                   SUBST    1
         DCL CONT   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));               SUBST    2
```

```
        DCL STRIND ENTRY(FIXED BIN(31,0),FIXED BIN(31,0))                          SUBST    3
                   RETURNS(FIXED BIN(31,0));                                       SUBST    4
        DCL CAL FIXED BIN(31,0) STATIC;                                            SUBST    5
        DCL SETIND ENTRY(FIXED BIN(31,0),FIXED BIN(31,0),                          SUBST    6
                         FIXED BIN(31,0),FIXED BIN(31,0))                          SUBST    7
                   RETURNS(FIXED BIN(31,0));                                       SUBST    8
        DCL LCNTR  ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));                SUBST    9
        DCL SUB FIXED BIN(31,0) STATIC;                                            SUBST   10
        DCL NAMTST ENTRY(FIXED BIN(31,0))RETURNS(FIXED BIN(31,0));                 SUBST   11
        SUB=CONT(N+1);                                                             SUBST   12
        CAL=STRIND(D,N+1);                                                         SUBST   13
        CALL SETIND(0,-1,-1,N);                                                    SUBST   14
        IF NAMTST(D)=-1 THEN GOTO LA;                                              SUBST   15
        CALL SETIND(1,-1,-1,N);                                                    SUBST   16
        CALL SETIND(-1,-1,LCNTR(D)+1,D+1);                                         SUBST   17
   LA:  IF NAMTST(SUB)=0 THEN CAL= IRALST(SUB);                                    SUBST   18
        RETURN(SUB);                                                               SUBST   19
        END SUBST;                                                                 SUBST   20


SUBSTP: PROC(DAT,LST)       FIXED BIN(31,0);                                       SUBSTP   0
        DCL (DAT,LST)       FIXED BIN(31,0);                                       SUBSTP   1
        DCL CONT   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));                SUBSTP   2
        DCL LNKR   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));                SUBSTP   3
        DCL SUBST ENTRY(FIXED BIN(31,0),FIXED BIN(31,0))                           SUBSTP   4
             RETURNS(FIXED BIN(31,0));                                             SUBSTP   5
        RETURN(SUBST(DAT,LNKR(CONT(LST))));                                        SUBSTP   6
        END SUBSTP;                                                                SUBSTP   7


TOP:                                                                               TOP      0
T:      PROC(P)             FIXED BIN(31,0);                                       TOP      1
```

```
DCL (P)              FIXED BIN(31,0);                                    TOP      2
DCL CONT   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));              TOP      3
DCL LNKR   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));              TOP      4
DCL LOCT   ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));              TOP      5
RETURN(CONT(LNKR(CONT(LOCT(P)))+1));                                     TOP      6
END TOP;                                                                 TOP      7
```

```
INHALT   CSECT                                                  INHALT   0
         EXTRN RELOC                                            INHALT   1
         SAVE  (1,4)                                            INHALT   2
         BALR  4,0                                              INHALT   3
         USING *,4                                              INHALT   4
         L     2,4(1)                                           INHALT   5
         L     1,0(1)          ADDR OF PARAM TO R1              INHALT   6
         L     1,0(1)         CONT OF ADDR TO R1                INHALT   7
         SLL   1,17                                             INHALT   8
         SRL   1,15                                             INHALT   9
         L     3,ADCON                                          INHALT  10
         A     1,0(3)                                           INHALT  11
         MVC   0(4,2),0(1)                                      INHALT  12
         RETURN (1,4)                                           INHALT  13
ADCON    DC    A(RELOC)                                         INHALT  14
         END                                                    INHALT  15


SETIND   CSECT                                                  SETIND   0
         EXTRN RELOC                                            SETIND   1
         SAVE  (2,7)                                            SETIND   2
         BALR  2,0                                              SETIND   3
         USING *,2                                              SETIND   4
         L     7,12(1)       ADDR OF CELL TO R7                 SETIND   5
         L     7,0(7)        ACTUAL ADDR OF CELL                SETIND   6
         SLL   7,17                                             SETIND   7
         SRL   7,15                                             SETIND   8
         L     3,ADCON                                          SETIND   9
         A     7,0(3)                                           SETIND  10
         L     3,0(7)        IND CONT OF CELL TO R3             SETIND  11
```

```
A0      LAVE   4,8(1)                                          SETIND  12
        LALR   4,C(4)          LNKR TO R4                      SETIND  13
        LSING  6,=X'FFFFFFFF'                                  SETIND  14
        CLR    4,6                                             SETIND  15
        BC     8,A1            BRANCH ON ZERO                  SETIND  16
        SRDL   4,15            SHIFT NEW LNKR TO R5            SETIND  17
        B      A2                                              SETIND  18
A1      LR     4,3                                             SETIND  19
        SRDL   4,15            SHIFT OLD LNKR TO R5            SETIND  20
A2      L      4,4(1)                                          SETIND  21
        L      4,C(4)          LNKL TO R4                      SETIND  22
        SRL    3,15                                            SETIND  23
        CLR    4,6                                             SETIND  24
        BC     8,A3                                            SETIND  25
        SRDL   4,15            NEW LNKL TO R5                  SETIND  26
        B      A4                                              SETIND  27
A3      LR     4,3                                             SETIND  28
        SRDL   4,15            OLD LNKL TO R5                  SETIND  29
A4      L      4,0(1)                                          SETIND  30
        L      4,0(4)          LLNKL TO R4                     SETIND  31
        SRL    3,15                                            SETIND  32
        CLR    4,6                                             SETIND  33
        BC     8,A5            IF ID = -1 THEN GOTO A5         SETIND  34
        SRDL   4,2             NEW ID TO R5                    SETIND  35
        B      A6                                              SETIND  36
A5      LR     4,3                                             SETIND  37
        SRDL   4,2                                             SETIND  38
A6      ST     5,0(7)                                          SETIND  39
        RETURN (2,7)                                           SETIND  40
ADCON   DC     A(RELOC)                                        SETIND  41
        END                                                    SETIND  42

STRDIR  CSECT                                                  STRDIR   0
```

```
         SAVE  (1,4)                                                    STRDIR   1
         BALR  2,0                                                      STRDIR   2
         USING *,2                                                      STRDIR   3
         L     4,8(1)                                                   STRDIR   4
         L     3,0(1)       ADDR OD DATUM TO R3                         STRDIR   5
         L     1,4(1)      ADDR OF CELL TO R1                           STRDIR   6
         MVC   0(4,1),0(3)   STORE DATUM IN CELL                        STRDIR   7
         MVC   0(4,4),0(3)   RETURN DATUM AS FUNCTION VALUE             STRDIR   8
         RETURN (1,4)                                                   STRDIR   9
         END                                                            STRDIR  10


SETDIR   CSECT                                                          SETDIR   0
         SAVE  (2,7)                                                    SETDIR   1
         BALR  2,0                                                      SETDIR   2
         USING *,2                                                      SETDIR   3
         L     7,12(1)        ADDR OF CELL TO R7                        SETDIR   4
         L     3,0(7)         CONT OF CELL TO R3                        SETDIR   5
A0       L     4,8(1)                                                   SETDIR   6
         L     4,0(4)         LNKR TO R4                                SETDIR   7
         L     6,=X'FFFFFFFF'                                           SETDIR   8
         CLR   4,6                                                      SETDIR   9
         BC    8,A1            BRANCH ON ZERO                          SETDIR  10
         SRDL  4,15            SHIFT NEW LNKR TO R5                    SETDIR  11
         B     A2                                                       SETDIR  12
A1       LR    4,3                                                      SETDIR  13
         SRDL  4,15            SHIFT OLD LNKR TO R5                    SETDIR  14
A2       L     4,4(1)                                                   SETDIR  15
         L     4,0(4)         LNKL TO R4                                SETDIR  16
         SRL   3,15                                                     SETDIR  17
         CLR   4,6                                                      SETDIR  18
         BC    8,A3                                                     SETDIR  19
         SRDL  4,15            NEW LNKL TO R5                          SETDIR  20
```

```
         B       A4                                                    SETDIR  21
A3       LR      4,3                                                   SETDIR  22
         SRDL    4,15            OLD LNKL TO R5                        SETDIR  23
A4       L       4,0(1)                                                SETDIR  24
         L       4,0(4)          LLNKL TO R4                           SETDIR  25
         SRL     3,15                                                  SETDIR  26
         CLR     4,6                                                   SETDIR  27
         BC      8,A5            IF ID = -1 THEN GOTO A5               SETDIR  28
         SRDL    4,2             NEW ID TO R5                          SETDIR  29
         B       A6                                                    SETDIR  30
A5       LR      4,3                                                   SETDIR  31
         SRDL    4,2                                                   SETDIR  32
A6       ST      5,0(7)                                                SETDIR  33
         RETURN (2,7)                                                  SETDIR  34
         END                                                           SETDIR  35


STRIND   CSECT                                                         STRIND   0
         EXTRN RELOC                                                   STRIND   1
         SAVE  (1,5)                                                   STRIND   2
         BALR  2,0                                                     STRIND   3
         USING *,2                                                     STRIND   4
         L     4,8(1)                                                  STRIND   5
         L     3,0(1)      ADDR OF DATUM TO R3                         STRIND   6
         L     1,4(1)      ADDR OF CELL TO R1                          STRIND   7
         L     1,0(1)      ADDR IN CELL TO R1                          STRIND   8
         SLL   1,17                                                    STRIND   9
         SRL   1,15                                                    STRIND  10
         L     5,ADCON                                                 STRIND  11
         A     1,0(5)                                                  STRIND  12
         MVC   0(4,1),0(3)   STORE DATUM IN LOC WHOSE ADDR IS IN CELL  STRIND  13
         MVC   0(4,4),0(3)   RETURN DATUM AS FUNCTION VALUE            STRIND  14
         RETURN (1,5)                                                  STRIND  15
```

```
ADCON    DC     A(RELOC)                                    STRIND  16
         END                                                STRIND  17


ID       CSECT                                              ID      0
         SAVE   (1,2)                                       ID      1
         L      2,4(1)                                      ID      2
          L     1,0(1)                                      ID      3
         L      1,0(1)                                      ID      4
         SRL    1,30                                        ID      5
         ST     1,0(2)                                      ID      6
         RETURN (1,2)                                       ID      7
         END                                                ID      8


LNKR     CSECT                                              LNKR    0
         SAVE   (1,2)                                       LNKR    1
         L      2,4(1)                                      LNKR    2
         L      1,0(1)                                      LNKR    3
         L      1,0(1)                                      LNKR    4
         SLL    1,17                                        LNKR    5
         SRL    1,17                                        LNKR    6
         ST     1,0(2)                                      LNKR    7
         RETURN (1,2)                                       LNKR    8
         END                                                LNKR    9


LNKL     CSECT                                              LNKL    0
         SAVE   (1,2)                                       LNKL    1
         L      2,4(1)                                      LNKL    2
         L      1,0(1)                                      LNKL    3
```

```
         L       1,0(1)                                          LNKL     4
         SLL     1,2                                             LNKL     5
         SRL     1,17                                            LNKL     6
         ST      1,0(2)                                          LNKL     7
         RETURN (1,2)                                            LNKL     8
         END                                                     LNKL     9


MADOV    CSECT                                                   MADOV    0
         SAVE   (1,2)                                            MADOV    1
         L       2,4(1)                                          MADOV    2
         L       1,0(1)                                          MADOV    3
         ST      1,0(2)                                          MADOV    4
         RETURN (1,2)                                            MADOV    5
         END                                                     MADOV    6


CONT     CSECT                                                   CONT     0
         EXTRN RELOC                                             CONT     1
         SAVE   (1,4)                                            CONT     2
         BALR    4,0                                             CONT     3
         USING *,4                                               CONT     4
         L       2,4(1)                                          CONT     5
         L       1,0(1)          ADDR OF PARAM TO R1             CONT     6
         L       1,0(1)         CONT OF ADDR TO R1               CONT     7
         SLL     1,17                                            CONT     8
         SRL     1,15                                            CONT     9
         L       3,ADCON                                         CONT    10
         A       1,0(3)                                          CONT    11
         MVC     0(4,2),0(1)                                     CONT    12
         RETURN (1,4)                                            CONT    13
ADCON    DC      A(RELOC)                                        CONT    14
```

```
      END                                                          CONT    15
```

```
CANONCL: PROC(STR,MOD)        RECURSIVE;                                    CANONCL  0
        /* PRNT IS 1 IF PRINTING OF INTERMEDIATE STAGES IS WANTED */        CANONCL  1
        /* BETA IS THE PRINCIPAL STRUCTURE */                               CANONCL  2
        DCL (                                                               CANONCL  3
         MADNTP,SUBST,SUBSBT,INLSTR,NXTLFT                                  CANONCL  4
        )ENTRY(FIXED BIN(31,0),FIXED BIN(31,0))                             CANONCL  5
         RETURNS(FIXED BIN(31,0));                                          CANONCL  6
        DCL (                                                               CANONCL  7
         CAL,L,LR,R,RR,MOD,STR,A,LA                                         CANONCL  8
        )     FIXED BIN(31,0);                                              CANONCL  9
        DCL (                                                               CANONCL 10
         SYM,SR,SL                                                          CANONCL 11
        )     CHAR(1);                                                      CANONCL 12
        DCL(                                                                CANONCL 13
        CANONCL,RINVOUT,LINVOUT,RPRDOUT,LPRDOUT,JACOBI,ORDPROD,             CANONCL 14
         STRPN,LEO,REO,EXPRD                                                CANONCL 15
         )ENTRY(FIXED BIN(31,0),FIXED BIN(31,0));                           CANONCL 16
        DCL (                                                               CANONCL 17
         SYMB                                                               CANONCL 18
        )ENTRY(FIXED BIN(31,0)) RETURNS(CHAR(1));                           CANONCL 19
        DCL (                                                               CANONCL 20
         IDNT,NTERM,COMCHK                                                  CANONCL 21
        )ENTRY(FIXED BIN(31,0)) RETURNS(BIT(1));                            CANONCL 22
        DCL (                                                               CANONCL 23
         PERMCOM,INVPRD,INVINV,ORPIDNT,CANCEL,                              CANONCL 24
         SUMEX,EXPINV,INVEX,PRODEXP,SUMINT,PRODINT                          CANONCL 25
        )ENTRY(FIXED BIN(31,0));                                            CANONCL 26
        DCL (                                                               CANONCL 27
         BOT,CONT,WT,MTLIST,TOP,LNKR                                        CANONCL 28
        )ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));                   CANONCL 29
        DCL (                                                               CANONCL 30
```

```
      EQ,LT                                                              CANONCL 31
     )ENTRY(FIXED BIN(31,0),FIXED BIN(31;0))                             CANONCL 32
      RETURNS(BIT(1));                                                   CANONCL 33
DCL (                                                                    CANONCL 34
      LG                                                                 CANONCL 35
     )ENTRY(CHAR(1),FIXED BIN(31,0),FIXED BIN(31,0))                     CANONCL 36
      RETURNS(FIXED BIN(31,0));                                          CANONCL 37
DCL SCAN ENTRY(FIXED BIN(31,0),FIXED BIN(31,0),                          CANONCL 38
              FIXED BIN(31,0),FIXED BIN(31,0));                          CANONCL 39
DCL PRNT FIXED BIN(31,0) EXT;                                            CANONCL 40
DCL BETA FIXED BIN(31,0) EXT;                                            CANONCL 41
DCL DI FIXED DEC;                                                        CANONCL 42
DCL C3 CHAR(3);                                                          CANONCL 43
DCL CT CHAR(2),DT FIXED DEC;                                             CANONCL 44
DCL JJJ STATIC INITIAL(0);                                               CANONCL 45
IF PRNT=1 THEN CALL STRPN(BETA,0);                                       CANONCL 46
IF PRNT=1 THEN JJJ=1;                                                    CANONCL 47
JJJ=JJJ+1; IF JJJ>50 THEN                                                CANONCL 48
 DO; JJJ=0; CALL STRPN(BETA,0); END;                                     CANONCL 49
IF ¬NTERM(STR) THEN RETURN;                                              CANONCL 50
L=CONT(MADNTP(STR,2)+1); R=BOT(STR); SYM=SYMB(STR);                      CANONCL 51
                                                                         CANONCL 52
                                                                         CANONCL 53
IF SYM='''' THEN DO;                                                     CANONCL 54
   IF¬NTERM(R) THEN DO;                                                  CANONCL 55
      IF IDNT(R) THEN DO; CALL CANCEL(STR); GOTO RET; END;END;           CANONCL 56
   SR=SYMB(R);                                                           CANONCL 57
   IF SR='.' THEN DO;                                                    CANONCL 58
      CALL INVPRD(STR);                                                  CANONCL 59
      CALL CANONCL(CONT(MADNTP(STR,2)+1),MOD);                           CANONCL 60
      CALL CANONCL(BOT(STR),MOD); GOTO RET;  END;                        CANONCL 61
   IF SR='''' THEN DO; CALL INVINV(STR); GOTO RET; END;                  CANONCL 62
   IF SR=':' THEN DO; CALL INVEX(STR); GOTO RET; END;                    CANONCL 63
   RETURN;                                                               CANONCL 64
```

```
                END; /*''''*/                                                    CANONCL 65
                                                                                 CANONCL 66
                                                                                 CANONCL 67
      IF SYM='.' THEN DO;                                                        CANONCL 68
         IF NTERM(L) THEN                                                        CANONCL 69
            IF SYMB(L)='''' & BOT(L)=R THEN                                      CANONCL 70
               DO; CALL CANCEL(STR); GOTO RET; END;                              CANONCL 71
         IF NTERM(R) THEN                                                        CANONCL 72
            IF SYMB(R)='''' & BOT(R)=L THEN                                      CANONCL 73
               DO; CALL CANCEL(STR); GOTO RET; END;                              CANONCL 74
         IF ¬NTERM(L) & IDNT(L) THEN                                             CANONCL 75
            DO; CALL DRPIDNT(STR); GOTO RET; END;                                CANONCL 76
         IF ¬NTERM(R) & IDNT(R) THEN                                             CANONCL 77
            DO; CALL DRPIDNT(STR); GOTO RET; END;                                CANONCL 78
         CALL ORDPROD(STR,MOD); RETURN;                                          CANONCL 79
         END; /*'.'*/                                                            CANONCL 80
                                                                                 CANONCL 81
                                                                                 CANONCL 82
      IF SYM=',' THEN DO;                                                        CANONCL 83
         IF IDNT(L) | IDNT(R) THEN DO;                                           CANONCL 84
            CAL=CANCEL(STR); GOTO RET; END;                                      CANONCL 85
         IF SYMB(L)=':' & SYMB(R)=':' THEN                                       CANONCL 86
            IF EQ(L,R) THEN DO; CAL=CANCEL(STR); GOTO RET; END;                  CANONCL 87
         IF SYMB(L)=',' & SYMB(R)=',' THEN                                       CANONCL 88
            IF EQ(L,R) THEN DO; CAL=CANCEL(STR); GOTO RET; END;                  CANONCL 89
         IF SYMB(L)='''' THEN DO;                                                CANONCL 90
            CAL=LINVOUT(STR,MOD); GOTO RES; END;                                 CANONCL 91
         IF SYMB(R)='''' THEN DO;                                                CANONCL 92
            CAL=RINVOUT(STR,MOD); GOTO RES; END;                                 CANONCL 93
         IF SYMB(L)='.' THEN DO;                                                 CANONCL 94
            CAL=LPRDOUT(STR,MOD); GOTO RET; END;                                 CANONCL 95
         IF SYMB(R)='.' THEN DO;                                                 CANONCL 96
            CAL=RPRDOUT(STR,MOD); GOTO RET; END;                                 CANONCL 97
         IF SYMB(R)=':' THEN DO;                                                 CANONCL 98
```

```
      CALL REO(STR,MOD); GOTO RET; END;                                  CANONCL 99
   IF SYMB(L)=':' THEN DO;                                               CANONCL100
      CALL LEO(STR,MOD); GOTO RET; END;                                  CANONCL101
   IF COMCHK(L) & COMCHK(R) & LT(L,R) THEN DO;                           CANONCL102
      CAL=PERMCOM(STR); GOTO RET; END;                                   CANONCL103
   LR=CONT(MADNTP(L,2)+1); SR=SYMB(L); RR=BOT(L);                        CANONCL104
   IF SR¬=',' THEN RETURN;                                               CANONCL105
   IF COMCHK(L) & COMCHK(R) & COMCHK(LR)                                 CANONCL106
      & LT(R,RR) & LT(RR,LR) THEN DO;                                    CANONCL107
      CALL JACOBI(STR,MOD); GOTO RES; END;                               CANONCL108
   RETURN;                                                               CANONCL109
      END; /*',*/                                                        CANONCL110
                                                                         CANONCL111
                                                                         CANONCL112
IF SYM=':' THEN DO;                                                      CANONCL113
   SR=SYMB(L);                                                           CANONCL114
   IF SR='''' THEN DO; CALL EXPINV(STR); GOTO RES;  END;                 CANONCL115
   IF SR=':' THEN DO; CALL PRODEXP(STR); GOTO RES;  END;                 CANONCL116
   IF SR='.' THEN DO;                                                    CANONCL117
      RR=TOP(R); UNSPEC(SL)=SUBSTR(UNSPEC(RR),9,8);                      CANONCL118
      IF SUBSTR(UNSPEC(RR),9,24)=UNSPEC('DOU') THEN DO;                  CANONCL119
         IF SUBSTR(UNSPEC(BOT(BOT(R))),9,8)¬=UNSPEC('I')                 CANONCL120
            THEN GOTO K2;                                                CANONCL121
         UNSPEC(CT)=SUBSTR(UNSPEC(BOT(BOT(R))),17,16);                   CANONCL122
         DI=CT; L,LR=LG('''',L,0);  GOTO K1;                             CANONCL123
                        END;                                             CANONCL124
      IF SL='I' THEN DO;                                                 CANONCL125
         UNSPEC(CT)=SUBSTR(UNSPEC(RR),17,16);                            CANONCL126
         DI=CT; LR=L;  GOTO K1;                                          CANONCL127
                        END;                                             CANONCL128
K2:   CALL EXPRD(STR,MOD); GOTO RET;                                     CANONCL129
K1:   DO I=2 TO DI; LR=LG('.',LR,L); END;                                CANONCL130
      CAL=IRALST(INLSTR(LR,MTLIST(STR)));                                CANONCL131
      GOTO RES;  END;                                                    CANONCL132
```

```
      C3=SUBSTR(UNSPEC(TOP(R)),9,24);                                          CANONCL133
      IF C3='I 0' | C3='I00' THEN DO; CAL=CANCEL(STR);  END;                   CANONCL134
      RETURN;  END;                                                            CANONCL135
                                                                               CANONCL136
                                                                               CANONCL137
   IF SYM='+' | SYM='-' THEN DO;                                               CANONCL138
      CALL SUMINT(STR); RETURN; END;                                           CANONCL139
                                                                               CANONCL140
                                                                               CANONCL141
   IF SYM='U' THEN DO;                                                         CANONCL142
      IF SYMB(L)='+'  | SYMB(L)='-' THEN DO;                                   CANONCL143
         CALL SUMINT(L); GOTO RET;  END;                                       CANONCL144
      IF ¬NTERM(L) THEN RETURN;                                                CANONCL145
      IF SYMB(L)¬='U' THEN RETURN;                                             CANONCL146
      A=BOT(L); CAL=MTLIST(STR);                                               CANONCL147
      LR=LNKR(A); LA=LNKR(CONT(LR));                                           CANONCL148
      DO I=1 TO 4;                                                             CANONCL149
         CAL=NXTLFT(CONT(MADNTP(A,I)+1),STR);                                  CANONCL150
         LA=LNKR(CONT(LA)); IF LA=LR THEN GOTO RET;                            CANONCL151
      END;                                                                     CANONCL152
                  END;                                                         CANONCL153
                                                                               CANONCL154
                                                                               CANONCL155
   IF SYM='*' THEN DO; CALL PRODINT(STR); RETURN; END;                         CANONCL156
   IF SYM='|' THEN RETURN;                                                     CANONCL157
                                                                               CANONCL158
                                                                               CANONCL159
   SIGNAL CONDITION(ALPHA);                                                    CANONCL160
                                                                               CANONCL161
RET: IF PRNT=1 THEN CALL STRPN(BETA,0); RETURN;                                CANONCL162
RES: IF PRNT=1 THEN CALL STRPN(BETA,0);                                        CANONCL163
   CALL SCAN(STR,STR,0,MOD);                                                   CANONCL164
   RETURN;                                                                     CANONCL165
                                                                               CANONCL166
```

```
        END CANONCL;                                                    CANONCL167

CMUTPRD: PROC(STR);                                                     CMUTPRD  0
        /*  AB -> BA(A,B)   */                                          CMUTPRD  1
        DCL (                                                           CMUTPRD  2
        CAL                                                             CMUTPRD  3
            ) BUT    FIXED BIN(31,0);                                   CMUTPRD  4
        DCL (                                                           CMUTPRD  5
            A,B,L,STR                                                   CMUTPRD  6
            )        FIXED BIN(31,0);                                   CMUTPRD  7
        DCL (                                                           CMUTPRD  8
             BOT,CONT,DELETE                                            CMUTPRD  9
            )ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));           CMUTPRD 10
        DCL (                                                           CMUTPRD 11
             MADNTP                                                     CMUTPRD 12
            )ENTRY(FIXED BIN(31,0),FIXED BIN(31,0))                     CMUTPRD 13
             RETURNS(FIXED BIN(31,0));                                  CMUTPRD 14
        DCL (                                                           CMUTPRD 15
             LG                                                         CMUTPRD 16
            )ENTRY(CHAR(1),FIXED BIN(31,0),FIXED BIN(31,0))             CMUTPRD 17
             RETURNS(FIXED BIN(31,0));                                  CMUTPRD 18
        A=CONT(MADNTP(STR,2)+1);                                        CMUTPRD 19
        B=BOT(STR);                                                     CMUTPRD 20
        STR=LG('.',LG('.',B,A),LG(',',A,B));                            CMUTPRD 21
        END CMUTPRD;                                                    CMUTPRD 22


COMCHK:  PROC(COM)          BIT(1)    RECURSIVE;                        COMCHK   0
        /* RETURNS '1'B IFF COM IS A COMMUTATOR */                      COMCHK   1
        DCL (                                                           COMCHK   2
             SYMB                                                       COMCHK   3
```

```
     )ENTRY(FIXED BIN(31,0)) RETURNS(CHAR(1));                          COMCHK   4
DCL (                                                                   COMCHK   5
     GRPL,COMCHK                                                        COMCHK   6
     )ENTRY(FIXED BIN(31,0)) RETURNS(BIT(1));                           COMCHK   7
DCL (                                                                   COMCHK   8
     COM                                                                COMCHK   9
     )     FIXED BIN(31,0);                                             COMCHK  10
DCL (                                                                   COMCHK  11
     BOT,CONT                                                           COMCHK  12
     )ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));                  COMCHK  13
DCL (                                                                   COMCHK  14
     MADNTP                                                             COMCHK  15
     )ENTRY(FIXED BIN(31,0),FIXED BIN(31,0))                            COMCHK  16
     RETURNS(FIXED BIN(31,0));                                          COMCHK  17
IF GRPL(COM) THEN RETURN('1'B);                                         COMCHK  18
IF SYMB(COM)¬=',',&SYMB(COM)¬='''' THEN RETURN('0'B);                   COMCHK  19
IF SYMB(COM)='''' THEN RETURN(COMCHK(BOT(COM)));                        COMCHK  20
RETURN(COMCHK(BOT(COM))&COMCHK(CONT(MADNTP(COM,2)+1)));                 COMCHK  21
END COMCHK;                                                             COMCHK  22


COMCNT: PROC(STR)          CHAR(1)   RECURSIVE;                         COMCNT   0
/* PROC ENTERS COMMA COUNTS IN THE STRUCTURE */                         COMCNT   1
DCL (                                                                   COMCNT   2
     A,B,STR,WTA,WTB,CAL                                                COMCNT   3
     )     FIXED BIN(31,0);                                             COMCNT   4
DCL (                                                                   COMCNT   5
     COMCNT                                                             COMCNT   6
     )ENTRY(FIXED BIN(31,0)) RETURNS(CHAR(1));                          COMCNT   7
DCL (                                                                   COMCNT   8
     SYMB                                                               COMCNT   9
     )ENTRY(FIXED BIN(31,0)) RETURNS(CHAR(1));                          COMCNT  10
DCL (                                                                   COMCNT  11
```

```
          MRKA,MRKB                                                          COMCNT  12
         )     CHAR(1);                                                      COMCNT  13
       DCL (                                                                 COMCNT  14
          SUBSTP,MADNTP                                                      COMCNT  15
         )ENTRY(FIXED BIN(31,0),FIXED BIN(31,0))                             COMCNT  16
          RETURNS(FIXED BIN(31,0));                                          COMCNT  17
       DCL (                                                                 COMCNT  18
          WT,CONT,BOT,TOP                                                    COMCNT  19
         )ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));                   COMCNT  20
       DCL MRKC CHAR(4);                                                     COMCNT  21
      DCL MRK(4) PACKED CHAR(1) DEFINED MRKC ;                               COMCNT  22
       DCL (                                                                 COMCNT  23
          NTERM                                                              COMCNT  24
         )ENTRY(FIXED BIN(31,0)) RETURNS(BIT(1));                            COMCNT  25
       DCL INK ENTRY(CHAR(1))RETURNS(CHAR(1));                               COMCNT  26
       ON CONVERSION SIGNAL CONDITION (ALPHA);                               COMCNT  27
       IF ¬NTERM(STR) THEN RETURN('0');                                      COMCNT  28
       A=CONT(MADNTP(STR,2)+1);                                              COMCNT  29
       B=BOT(STR);                                                           COMCNT  30
       UNSPEC(MRKC)=UNSPEC(TOP(STR));                                        COMCNT  31
       MRKA=COMCNT(A); MRKB=COMCNT(B);                                       COMCNT  32
       IF SYMB(STR)¬=',' THEN RETURN('0');                                   COMCNT  33
       IF MRKA >MRKB THEN MRK(3)=MRKA; ELSE MRK(3)=INK(MRKB);                COMCNT  34
       CAL=SUBSTP(UNSPEC(MRKC),STR); RETURN(MRK(3));                         COMCNT  35
       END COMCNT;                                                           COMCNT  36


COPYPRD: PROC(STR) FIXED BIN(31,0)     RECURSIVE;                            COPYPRD  0
                                                                             COPYPRD  1
       DCL STR FIXED BIN(31,0);                                              COPYPRD  2
                                                                             COPYPRD  3
       DCL (                                                                 COPYPRD  4
                                                                             COPYPRD  5
```

```
      LG                                                          COPYPRD  6
     )ENTRY(CHAR(1),FIXED BIN(31,0),FIXED BIN(31,0))              COPYPRD  7
      RETURNS(FIXED BIN(31,0));                                   COPYPRD  8
                                                                  COPYPRD  9
   DCL (                                                          COPYPRD 10
                                                                  COPYPRD 11
CONT,BOT,COPYPRD                                                  COPYPRD 12
                                                                  COPYPRD 13
     )ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));            COPYPRD 14
                                                                  COPYPRD 15
   DCL (                                                          COPYPRD 16
                                                                  COPYPRD 17
      MADNTP                                                      COPYPRD 18
                                                                  COPYPRD 19
     )ENTRY(FIXED BIN(31,0),FIXED BIN(31,0))                      COPYPRD 20
      RETURNS(FIXED BIN(31,0));                                   COPYPRD 21
                                                                  COPYPRD 22
   DCL (                                                          COPYPRD 23
                                                                  COPYPRD 24
      SYMB                                                        COPYPRD 25
                                                                  COPYPRD 26
     )ENTRY(FIXED BIN(31,0)) RETURNS(CHAR(1));                    COPYPRD 27
                                                                  COPYPRD 28
   DCL (                                                          COPYPRD 29
                                                                  COPYPRD 30
      NTERM                                                       COPYPRD 31
                                                                  COPYPRD 32
     )ENTRY(FIXED BIN(31,0)) RETURNS(BIT(1));                     COPYPRD 33
                                                                  COPYPRD 34
   IF ¬NTERM(STR) THEN RETURN(STR);                               COPYPRD 35
                                                                  COPYPRD 36
                                                                  COPYPRD 37
                                                                  COPYPRD 38
                                                                  COPYPRD 39
```

```
        IF SYMB(STR)¬='.' THEN RETURN(STR);                                COPYPRD 40
                                                                           COPYPRD 41
        RETURN(LG('.',COPYPRD(CONT(MADNTP(STR,2)+1))                       COPYPRD 42
                                                                           COPYPRD 43
                     ,COPYPRD(BOT(STR))));                                 COPYPRD 44
                                                                           COPYPRD 45
        END COPYPRD;                                                       COPYPRD 46


DRPIDNT: PROC(STR);                                                        DRPIDNT  0
        DCL(                                                               DRPIDNT  1
            CAL,STR,J,LA,LR,I                                              DRPIDNT  2
            ) FIXED BIN(31,0);                                             DRPIDNT  3
        DCL(                                                               DRPIDNT  4
            TOP,BOT,CONT,MTLIST,LNKR                                       DRPIDNT  5
            )ENTRY(FIXED BIN(31,0))RETURNS(FIXED BIN(31,0));               DRPIDNT  6
        DCL(                                                               DRPIDNT  7
        MADNTP,INLSTR,NXTLFT                                               DRPIDNT  8
            )ENTRY(FIXED BIN(31,0),FIXED BIN(31,0))                        DRPIDNT  9
            RETURNS(FIXED BIN(31,0));                                      DRPIDNT 10
        IF SUBSTR(UNSPEC(TOP(BOT(STR))),9,8)=UNSPEC('U')                   DRPIDNT 11
        THEN J=CONT(MADNTP(STR,2)+1);                                      DRPIDNT 12
        ELSE J=BOT(STR);                                                   DRPIDNT 13
        CAL=MTLIST(STR); LR=LNKR(J); LA=LNKR(CONT(LR));                    DRPIDNT 14
        DO I=1 TO 4;                                                       DRPIDNT 15
           CAL=NXTLFT(CONT(MADNTP(J,I)+1),STR);                            DRPIDNT 16
           LA=LNKR(CONT(LA)); IF LA=LR THEN RETURN;                        DRPIDNT 17
        END;                                                               DRPIDNT 18
        SIGNAL CONDITION(ALPHA);                                           DRPIDNT 19
        END DRPIDNT;                                                       DRPIDNT 20


EQ:     PROC(A,B)     BIT(1)  RECURSIVE;                                   EQ       0
```

```
        /* RETURNS '1'B IFF A=B */                                          EQ      1
        DCL (ANT,BNT) BIT(1);                                               EQ      2
        DCL (                                                               EQ      3
             MADNTP                                                         EQ      4
            )ENTRY(FIXED BIN(31,0),FIXED BIN(31,0))                         EQ      5
             RETURNS(FIXED BIN(31,0));                                      EQ      6
        DCL (                                                               EQ      7
             L,A,B                                                          EQ      8
            )     FIXED BIN(31,0);                                          EQ      9
        DCL (                                                               EQ     10
             NTERM                                                          EQ     11
            )ENTRY(FIXED BIN(31,0)) RETURNS(BIT(1));                        EQ     12
        DCL (                                                               EQ     13
             SYMB                                                           EQ     14
            )ENTRY(FIXED BIN(31,0)) RETURNS(CHAR(1));                       EQ     15
        DCL (                                                               EQ     16
             CONT,BOT                                                       EQ     17
            )ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));               EQ     18
        DCL (                                                               EQ     19
             EQ                                                             EQ     20
            )ENTRY(FIXED BIN(31,0),FIXED BIN(31,0))                         EQ     21
             RETURNS(BIT(1));                                               EQ     22
        DCL (SA,SB) CHAR(1);                                                EQ     23
        ANT=NTERM(A); BNT=NTERM(B);                                         EQ     24
        IF ¬ANT & ¬BNT THEN RETURN(SYMB(A)=SYMB(B));                        EQ     25
        SA=SYMB(A); SB=SYMB(B);                                             EQ     26
        IF SA=':' | SA='''' THEN RETURN(EQ(CONT(MADNTP(A,2)+1),B));         EQ     27
        IF SB=':' | SB='''' THEN RETURN(EQ(A,CONT(MADNTP(B,2)+1)));         EQ     28
        IF SA¬= SB THEN RETURN('0'B);                                       EQ     29
        RETURN(EQ(CONT(MADNTP(A,2)+1),CONT(MADNTP(B,2)+1))                  EQ     30
              &EQ(BOT(A),BOT(B)));                                          EQ     31
        END EQ;                                                             EQ     32


EXPINV:  PROC(STR)  FIXED BIN(31,0);                                        EXPINV  0
```

```
        DCL (                                                                   EXPINV   1
             STR,A,N,CAL                                                        EXPINV   2
            )     FIXED BIN(31,0);                                              EXPINV   3
        DCL (                                                                   EXPINV   4
             BOT,TOP,CONT,MTLIST                                                EXPINV   5
            )ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));                   EXPINV   6
        DCL (                                                                   EXPINV   7
             LG                                                                 EXPINV   8
            )ENTRY(CHAR(1),FIXED BIN(31,0),FIXED BIN(31,0))                     EXPINV   9
             RETURNS(FIXED BIN(31,0));                                          EXPINV  10
        DCL (                                                                   EXPINV  11
             MADNTP,INLSTR                                                      EXPINV  12
            )ENTRY(FIXED BIN(31,0),FIXED BIN(31,0))                             EXPINV  13
             RETURNS(FIXED BIN(31,0));                                          EXPINV  14
        DCL IRALST ENTRY(FIXED BIN(31,0));                                      EXPINV  15
        A=BOT(CONT(MADNTP(STR,2)+1));                                           EXPINV  16
        N=BOT(STR);                                                             EXPINV  17
        IF SUBSTR(UNSPEC(TOP(N)),9,24)=UNSPEC('00U')                            EXPINV  18
           THEN A=LG(':',A,BOT(N));                                             EXPINV  19
           ELSE A =LG(':',A,LG('U',N,0));                                       EXPINV  20
        CAL= IRALST(INLSTR(A,MTLIST(STR)));                                     EXPINV  21
        END EXPINV;                                                             EXPINV  22


EXPRD:  PROC(STR,MOD);                                                          EXPRD    0
        /* THIS PROCEDURE GENERATES ACANONIC EXPANSION OF                      EXPRD    1
          (A.B):N MOD WEIGHT 7 OR LESS                 */                     EXPRD    2
        DCL (                                                                   EXPRD    3
             STR,EX,L,LA,A,B,N,BA,BAA,BAB,BAAA,BAAB,BABB,BAAAA,                 EXPRD    4
             BAAAB,BAABB,BABBB,BAABA,BABBA,BAAAAA,BAAAAB,BAAABB,                EXPRD    5
             BAABBB,BABBBB,BAAABA,BAABBA,BABBBA,BABBAA,                         EXPRD    6
             MOD,BETS,CAL,WA,WB,                                                EXPRD    7
             I2,I3,I4,I5,I6,I7,I8,I9,I10,                                       EXPRD    8
```

```
     I12,I13,I18,I21,I20,I24,I27,I30,I36,I52,I54                    EXPRD    9
    )     FIXED BIN(31,0);                                          EXPRD   10
DCL (                                                               EXPRD   11
     LG                                                             EXPRD   12
    )ENTRY(CHAR(1),FIXED BIN(31,0),FIXED BIN(31,0))                 EXPRD   13
     RETURNS(FIXED BIN(31,0));                                      EXPRD   14
DCL (                                                               EXPRD   15
     MADNTP,INLSTR,NXTLFT                                           EXPRD   16
    )ENTRY(FIXED BIN(31,0),FIXED BIN(31,0))                         EXPRD   17
     RETURNS(FIXED BIN(31,0));                                      EXPRD   18
DCL (                                                               EXPRD   19
     WT,CONT,BOT,MTLIST,LIST                                        EXPRD   20
    )ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));               EXPRD   21
DCL PRNTI BIT(1) EXT;                                               EXPRD   22
DCL BETA FIXED BIN(31,0) EXT;                                       EXPRD   23
DCL (                                                               EXPRD   24
     CANONCL                                                        EXPRD   25
    )ENTRY(FIXED BIN(31,0),FIXED BIN(31,0));                        EXPRD   26
DCL(                                                                EXPRD   27
     STRPN                                                          EXPRD   28
   )ENTRY(FIXED BIN(31,0),FIXED BIN(31,0));                         EXPRD   29
IF PRNTI THEN PUT SKIP(2) LIST('ENTER EXPRD');                      EXPRD   30
L=CONT(MADNTP(STR,2)+1); N=BOT(STR);                                EXPRD   31
A=CONT(MADNTP(L,2)+1); B=BOT(L);                                    EXPRD   32
I2=LIST(9); CAL =NXTLFT(UNSPEC('0I02'),I2);                         EXPRD   33
I3=LIST(9); CAL =NXTLFT(UNSPEC('0I03'),I3);                         EXPRD   34
I4=LIST(9); CAL =NXTLFT(UNSPEC('0I04'),I4);                         EXPRD   35
I5=LIST(9); CAL =NXTLFT(UNSPEC('0I05'),I5);                         EXPRD   36
I6=LIST(9); CAL =NXTLFT(UNSPEC('0I06'),I6);                         EXPRD   37
I7=LIST(9); CAL =NXTLFT(UNSPEC('0I07'),I7);                         EXPRD   38
I8=LIST(9); CAL =NXTLFT(UNSPEC('0I08'),I8);                         EXPRD   39
I9=LIST(9); CAL =NXTLFT(UNSPEC('0I09'),I9);                         EXPRD   40
I10=LIST(9); CAL =NXTLFT(UNSPEC('0I10'),I10);                       EXPRD   41
I12=LIST(9); CAL =NXTLFT(UNSPEC('0I12'),I12);                       EXPRD   42
```

```
I13=LIST(9); CAL =NXTLFT(UNSPEC('0I13'),I13);                    EXPRD   43
I18=LIST(9); CAL =NXTLFT(UNSPEC('0I18'),I18);                    EXPRD   44
I20=LIST(9); CAL =NXTLFT(UNSPEC('0I20'),I20);                    EXPRD   45
I21=LIST(9); CAL =NXTLFT(UNSPEC('0I21'),I21);                    EXPRD   46
I24=LIST(9); CAL =NXTLFT(UNSPEC('0I24'),I24);                    EXPRD   47
I27=LIST(9); CAL =NXTLFT(UNSPEC('0I27'),I27);                    EXPRD   48
I30=LIST(9); CAL =NXTLFT(UNSPEC('0I30'),I30);                    EXPRD   49
I36=LIST(9); CAL =NXTLFT(UNSPEC('0I36'),I36);                    EXPRD   50
I52=LIST(9); CAL =NXTLFT(UNSPEC('0I52'),I52);                    EXPRD   51
I54=LIST(9); CAL =NXTLFT(UNSPEC('0I54'),I54);                    EXPRD   52
IF PRNTI THEN CALL STRPN(STR,0);                                 EXPRD   53
BETS=BETA;                                                       EXPRD   54
LA=LG(':',A,N);                                                  EXPRD   55
IF PRNTI THEN CALL STRPN(LA,0);                                  EXPRD   56
BETA=LA; CALL CANONCL(LA,MOD);                                   EXPRD   57
L=LG(':',B,N);                                                   EXPRD   58
IF PRNTI THEN CALL STRPN(L,0);                                   EXPRD   59
BETA=L; CALL CANONCL(L,MOD);                                     EXPRD   60
LA=LG('.',LA,L);                                                 EXPRD   61
IF PRNTI THEN CALL STRPN(LA,0);                                  EXPRD   62
BETA=LA; CALL CANONCL(LA,MOD);                                   EXPRD   63
WA=WT(A); WB=WT(B);                                              EXPRD   64
                                                                 EXPRD   65
IF WA+WB< MOD THEN DO;                                           EXPRD   66
   BA=LG(',',B,A);                                               EXPRD   67
   IF PRNTI THEN CALL STRPN(BA     ,0);                          EXPRD   68
   BETA=BA     ; CALL CANONCL(BA     ,MOD);                      EXPRD   69
   EX=LG('|',N,I2);                                              EXPRD   70
   L=LG(':',BA,EX);                                              EXPRD   71
   IF PRNTI THEN CALL STRPN(L,0);                                EXPRD   72
   BETA=L; CALL CANONCL(L,MOD);                                  EXPRD   73
   LA=LG('.',LA,L);                                              EXPRD   74
   IF PRNTI THEN CALL STRPN(LA,0);                               EXPRD   75
   BETA=LA; CALL CANONCL(LA,MOD); END;                           EXPRD   76
```

```
                                                                    EXPRD   77
IF 2*WA+WB< MOD THEN DO;                                            EXPRD   78
   BAA=LG(',',BA,A);                                                EXPRD   79
   IF PRNTI THEN CALL STRPN(BAA    ,0);                             EXPRD   80
   BETA=BAA    ; CALL CANONCL(BAA    ,MOD);                         EXPRD   81
   EX=LG('|',N,I3);                                                 EXPRD   82
   L=LG(':',BAA,EX);                                                EXPRD   83
   IF PRNTI THEN CALL STRPN(L,0);                                   EXPRD   84
   BETA=L; CALL CANONCL(L,MOD);                                     EXPRD   85
   LA=LG('.',LA,L);                                                 EXPRD   86
   IF PRNTI THEN CALL STRPN(LA,0);                                  EXPRD   87
   BETA=LA; CALL CANONCL(LA,MOD); END;                              EXPRD   88
                                                                    EXPRD   89
IF WA+2*WB < MOD THEN DO;                                           EXPRD   90
   BAB=LG(',',BA,B);                                                EXPRD   91
   IF PRNTI THEN CALL STRPN(BAB    ,0);                             EXPRD   92
   BETA=BAB    ; CALL CANONCL(BAB    ,MOD);                         EXPRD   93
   EX=LG('+',LG('|',N,I2),                                          EXPRD   94
      LG('*',I2,LG('|',N,I3)));                                     EXPRD   95
   L=LG(':',BAB,EX);                                                EXPRD   96
   IF PRNTI THEN CALL STRPN(L,0);                                   EXPRD   97
   BETA=L; CALL CANONCL(L,MOD);                                     EXPRD   98
   LA=LG('.',LA,L);                                                 EXPRD   99
   IF PRNTI THEN CALL STRPN(LA,0);                                  EXPRD  100
   BETA=LA; CALL CANONCL(LA,MOD); END;                              EXPRD  101
                                                                    EXPRD  102
IF 3*WA+WB <MOD THEN DO;                                            EXPRD  103
   BAAA=LG(',',BAA,A);                                              EXPRD  104
   IF PRNTI THEN CALL STRPN(BAAA   ,0);                             EXPRD  105
   BETA=BAAA   ; CALL CANONCL(BAAA   ,MOD);                         EXPRD  106
   EX=LG('|',N,I4);                                                 EXPRD  107
   L=LG(':',BAAA,EX);                                               EXPRD  108
   IF PRNTI THEN CALL STRPN(L,0);                                   EXPRD  109
   BETA=L; CALL CANONCL(L,MOD);                                     EXPRD  110
```

```
      LA=LG('.',LA,L);                                          EXPRD  111
      IF PRNTI THEN CALL STRPN(LA,0);                           EXPRD  112
      BETA=LA; CALL CANONCL(LA,MOD); END;                       EXPRD  113
                                                                EXPRD  114
   IF 2*WA+2*WB < MOD THEN DO;                                  EXPRD  115
      BAAB=LG(',',BAA,B);                                       EXPRD  116
      IF PRNTI THEN CALL STRPN(BAAB  ,0);                       EXPRD  117
      BETA=BAAB  ; CALL CANONCL(BAAB  ,MOD);                    EXPRD  118
      EX=LG('+',LG('*',I2,LG('|',N,I3)),                        EXPRD  119
              LG('*',I3,LG('|',N,I4)));                         EXPRD  120
      L=LG(':',BAAB,EX);                                        EXPRD  121
      IF PRNTI THEN CALL STRPN(L,0);                            EXPRD  122
      BETA=L; CALL CANONCL(L,MOD);                              EXPRD  123
      LA=LG('.',LA,L);                                          EXPRD  124
      IF PRNTI THEN CALL STRPN(LA,0);                           EXPRD  125
      BETA=LA; CALL CANONCL(LA,MOD); END;                       EXPRD  126
                                                                EXPRD  127
   IF WA+3*WB < MOD THEN DO;                                    EXPRD  128
      BABB=LG(',',BAB,B);                                       EXPRD  129
      IF PRNTI THEN CALL STRPN(BABB  ,0);                       EXPRD  130
      BETA=BABB  ; CALL CANONCL(BABB  ,MOD);                    EXPRD  131
      EX=LG('+',LG('*',I2,LG('|',N,I3)),                        EXPRD  132
              LG('*',I3,LG('|',N,I4)));                         EXPRD  133
      L=LG(':',BABB,EX);                                        EXPRD  134
      IF PRNTI THEN CALL STRPN(L,0);                            EXPRD  135
      BETA=L; CALL CANONCL(L,MOD);                              EXPRD  136
      LA=LG('.',LA,L);                                          EXPRD  137
      IF PRNTI THEN CALL STRPN(LA,0);                           EXPRD  138
      BETA=LA; CALL CANONCL(LA,MOD); END;                       EXPRD  139
                                                                EXPRD  140
   IF 4*WA+WB < MOD THEN DO;                                    EXPRD  141
      BAAAA=LG(',',BAAA,A);                                     EXPRD  142
      IF PRNTI THEN CALL STRPN(BAAAA ,0);                       EXPRD  143
      BETA=BAAAA ; CALL CANONCL(BAAAA ,MOD);                    EXPRD  144
```

```
   EX=LG('|',N,I5);                                                  EXPRD  145
   L=LG(':',BAAAA,EX);                                               EXPRD  146
   IF PRNTI THEN CALL STRPN(L,0);                                    EXPRD  147
   BETA=L; CALL CANONCL(L,MOD);                                      EXPRD  148
   LA=LG('.',LA,L);                                                  EXPRD  149
   IF PRNTI THEN CALL STRPN(LA,0);                                   EXPRD  150
   BETA=LA; CALL CANONCL(LA,MOD); END;                               EXPRD  151
                                                                     EXPRD  152
IF 3*WA+2*WB < MOD THEN DO;                                          EXPRD  153
   BAAAB=LG(',',BAAA,B);                                             EXPRD  154
   IF PRNTI THEN CALL STRPN(BAAAB ,0);                               EXPRD  155
   BETA=BAAAB ; CALL CANONCL(BAAAB ,MOD);                            EXPRD  156
   EX=LG('+',LG('*',I3,LG('|',N,I4)),                                EXPRD  157
            LG('*',I4,LG('|',N,I5)));                                EXPRD  158
   L=LG(':',BAAAB,EX);                                               EXPRD  159
   IF PRNTI THEN CALL STRPN(L,0);                                    EXPRD  160
   BETA=L; CALL CANONCL(L,MOD);                                      EXPRD  161
   LA=LG('.',LA,L);                                                  EXPRD  162
   IF PRNTI THEN CALL STRPN(LA,0);                                   EXPRD  163
   BETA=LA; CALL CANONCL(LA,MOD); END;                               EXPRD  164
                                                                     EXPRD  165
IF 2*WA+3*WB < MOD THEN DO;                                          EXPRD  166
   BAABB=LG(',',BAAB,B);                                             EXPRD  167
   IF PRNTI THEN CALL STRPN(BAABB ,0);                               EXPRD  168
   BETA=BAABB ; CALL CANONCL(BAABB ,MOD);                            EXPRD  169
   EX=LG('+',LG('|',N,I3),                                           EXPRD  170
      LG('+',LG('*',I6,LG('|',N,I4)),                                EXPRD  171
      LG('*',I6,LG('|',N,I5))));                                     EXPRD  172
   L=LG(':',BAABB,EX);                                               EXPRD  173
   IF PRNTI THEN CALL STRPN(L,0);                                    EXPRD  174
   BETA=L; CALL CANONCL(L,MOD);                                      EXPRD  175
   LA=LG('.',LA,L);                                                  EXPRD  176
   IF PRNTI THEN CALL STRPN(LA,0);                                   EXPRD  177
   BETA=LA; CALL CANONCL(LA,MOD); END;                               EXPRD  178
```

```
                                                                        EXPRD  179
IF WA+4*WB < MOD THEN DO;                                               EXPRD  180
   BABBB=LG(',',BABB,B);                                                EXPRD  181
   IF PRNTI THEN CALL STRPN(BABBB ,0);                                  EXPRD  182
   BETA=BABBB ; CALL CANONCL(BABBB ,MOD);                               EXPRD  183
   EX=LG('+',LG('*',I3,LG('|',N,I4)),                                   EXPRD  184
            LG('*',I4,LG('|',N,I5)));                                   EXPRD  185
   L=LG(':',BABBB,EX);                                                  EXPRD  186
   IF PRNTI THEN CALL STRPN(L,0);                                       EXPRD  187
   BETA=L; CALL CANONCL(L,MOD);                                         EXPRD  188
   LA=LG('.',LA,L);                                                     EXPRD  189
   IF PRNTI THEN CALL STRPN(LA,0);                                      EXPRD  190
   BETA=LA; CALL CANONCL(LA,MOD); END;                                  EXPRD  191
                                                                        EXPRD  192
IF 3*WA+2*WB < MOD THEN DO;                                             EXPRD  193
   BAABA=LG(',',BAA,BA);                                                EXPRD  194
   IF PRNTI THEN CALL STRPN(BAABA ,0);                                  EXPRD  195
   BETA=BAABA ; CALL CANONCL(BAABA ,MOD);                               EXPRD  196
   EX=LG('+',LG('|',N,I3),                                              EXPRD  197
      LG('+',LG('*',I7,LG('|',N,I4)),                                   EXPRD  198
      LG('*',I6,LG('|',N,I5))));                                        EXPRD  199
   L=LG(':',BAABA,EX);                                                  EXPRD  200
   IF PRNTI THEN CALL STRPN(L,0);                                       EXPRD  201
   BETA=L; CALL CANONCL(L,MOD);                                         EXPRD  202
   LA=LG('.',LA,L);                                                     EXPRD  203
   IF PRNTI THEN CALL STRPN(LA,0);                                      EXPRD  204
   BETA=LA; CALL CANONCL(LA,MOD); END;                                  EXPRD  205
                                                                        EXPRD  206
IF 2*WA+3*WB < MOD THEN DO;                                             EXPRD  207
   BABBA=LG(',',BAB,BA);                                                EXPRD  208
   IF PRNTI THEN CALL STRPN(BABBA ,0);                                  EXPRD  209
   BETA=BABBA ; CALL CANONCL(BABBA ,MOD);                               EXPRD  210
EX=LG('+',LG('*',I6,LG('|',N,I3)),LG('+',LG('*',                        EXPRD  211
   I18,LG('|',N,I4)),LG('*',I12,LG('|',N,I5))));                        EXPRD  212
```

```
      L=LG(':',BABBA,EX);                                          EXPRD  213
      IF PRNTI THEN CALL STRPN(L,0);                               EXPRD  214
      BETA=L; CALL CANONCL(L,MOD);                                 EXPRD  215
      LA=LG('.',LA,L);                                             EXPRD  216
      IF PRNTI THEN CALL STRPN(LA,0);                              EXPRD  217
      BETA=LA; CALL CANONCL(LA,MOD); END;                          EXPRD  218
                                                                   EXPRD  219
   IF 5*WA+WB < MOD THEN DO;                                       EXPRD  220
      BAAAAA=LG(',',BAAAA,A);                                      EXPRD  221
      IF PRNTI THEN CALL STRPN(BAAAAA,0);                          EXPRD  222
      BETA=BAAAAA; CALL CANONCL(BAAAAA,MOD);                       EXPRD  223
      EX=LG('|',N,I6);                                             EXPRD  224
      L=LG(':',BAAAAA,EX);                                         EXPRD  225
      IF PRNTI THEN CALL STRPN(L,0);                               EXPRD  226
      BETA=L; CALL CANONCL(L,MOD);                                 EXPRD  227
      LA=LG('.',LA,L);                                             EXPRD  228
      IF PRNTI THEN CALL STRPN(LA,0);                              EXPRD  229
      BETA=LA; CALL CANONCL(LA,MOD); END;                          EXPRD  230
                                                                   EXPRD  231
   IF 4*WA+2*WB < MOD THEN DO;                                     EXPRD  232
      BAAAAB=LG(',',BAAAA,B);                                      EXPRD  233
      IF PRNTI THEN CALL STRPN(BAAAAB,0);                          EXPRD  234
      BETA=BAAAAB; CALL CANONCL(BAAAAB,MOD);                       EXPRD  235
      EX=LG('+',LG('*',I4,LG('|',N,I5)),                           EXPRD  236
         LG('*',I5,LG('|',N,I6)));                                 EXPRD  237
      L=LG(':',BAAAAB,EX);                                         EXPRD  238
      IF PRNTI THEN CALL STRPN(L,0);                               EXPRD  239
      BETA=L; CALL CANONCL(L,MOD);                                 EXPRD  240
      LA=LG('.',LA,L);                                             EXPRD  241
      IF PRNTI THEN CALL STRPN(LA,0);                              EXPRD  242
      BETA=LA; CALL CANONCL(LA,MOD); END;                          EXPRD  243
                                                                   EXPRD  244
   IF 3*WA+3*WB < MOD THEN DO;                                     EXPRD  245
      BAAABB=LG(',',BAAAB,B);                                      EXPRD  246
```

```
      IF PRNTI THEN CALL STRPN(BAAABB,0);                              EXPRD  247
      BETA=BAAABB; CALL CANONCL(BAAABB,MOD);                           EXPRD  248
      EX=LG('+',LG('*',I3,LG('|',N,I4)),                               EXPRD  249
         LG('+',LG('*',I12,LG('|',N,I5)),                              EXPRD  250
                LG('*',I10,LG('|',N,I6))));                            EXPRD  251
      L=LG(':',BAAABB,EX);                                             EXPRD  252
      IF PRNTI THEN CALL STRPN(L,0);                                   EXPRD  253
      BETA=L; CALL CANONCL(L,MOD);                                     EXPRD  254
      LA=LG('.',LA,L);                                                 EXPRD  255
      IF PRNTI THEN CALL STRPN(LA,0);                                  EXPRD  256
      BETA=LA; CALL CANONCL(LA,MOD); END;                              EXPRD  257
                                                                       EXPRD  258
   IF 2*WA+4*WB < MOD THEN DO;                                         EXPRD  259
      BAABBB=LG(',',BAABB,B);                                          EXPRD  260
      IF PRNTI THEN CALL STRPN(BAABBB,0);                              EXPRD  261
      BETA=BAABBB; CALL CANONCL(BAABBB,MOD);                           EXPRD  262
      EX=LG('+',LG('*',I3,LG('|',N,I4)),                               EXPRD  263
         LG('+',LG('*',I12,LG('|',N,I5)),                              EXPRD  264
                LG('*',I10,LG('|',N,I6))));                            EXPRD  265
      L=LG(':',BAABBB,EX);                                             EXPRD  266
      IF PRNTI THEN CALL STRPN(L,0);                                   EXPRD  267
      BETA=L; CALL CANONCL(L,MOD);                                     EXPRD  268
      LA=LG('.',LA,L);                                                 EXPRD  269
      IF PRNTI THEN CALL STRPN(LA,0);                                  EXPRD  270
      BETA=LA; CALL CANONCL(LA,MOD); END;                              EXPRD  271
                                                                       EXPRD  272
   IF WA+5*WB < MOD THEN DO;                                           EXPRD  273
      BABBBB=LG(',',BABBB,B);                                          EXPRD  274
      IF PRNTI THEN CALL STRPN(BABBBB,0);                              EXPRD  275
      BETA=BABBBB; CALL CANONCL(BABBBB,MOD);                           EXPRD  276
      EX=LG('+',LG('*',I4,LG('|',N,I5)),                               EXPRD  277
         LG('*',I5,LG('|',N,I6)));                                     EXPRD  278
      L=LG(':',BABBBB,EX);                                             EXPRD  279
      IF PRNTI THEN CALL STRPN(L,0);                                   EXPRD  280
```

```
   BETA=L; CALL CANONCL(L,MOD);                                    EXPRD  281
   LA=LG('.',LA,L);                                                EXPRD  282
   IF PRNTI THEN CALL STRPN(LA,0);                                 EXPRD  283
   BETA=LA; CALL CANONCL(LA,MOD); END;                             EXPRD  284
                                                                   EXPRD  285
IF 4*WA+2*WB < MOD THEN DO;                                        EXPRD  286
   BAAABA=LG(',',BAAA,BA);                                         EXPRD  287
   IF PRNTI THEN CALL STRPN(BAAABA,0);                             EXPRD  288
   BETA=BAAABA; CALL CANONCL(BAAABA,MOD);                          EXPRD  289
   EX=LG('+',LG('*',I3,LG('|',N,I4)),                              EXPRD  290
      LG('+',LG('*',I13,LG('|',N,I5)),                             EXPRD  291
         LG('*',I10,LG('|',N,I6))));                               EXPRD  292
   L=LG(':',BAAABA,EX);                                            EXPRD  293
   IF PRNTI THEN CALL STRPN(L,0);                                  EXPRD  294
   BETA=L; CALL CANONCL(L,MOD);                                    EXPRD  295
   LA=LG('.',LA,L);                                                EXPRD  296
   IF PRNTI THEN CALL STRPN(LA,0);                                 EXPRD  297
   BETA=LA; CALL CANONCL(LA,MOD); END;                             EXPRD  298
                                                                   EXPRD  299
IF 3*WA+3*WB < MOD THEN DO;                                        EXPRD  300
   BAABBA=LG(',',BAAB,BA);                                         EXPRD  301
   IF PRNTI THEN CALL STRPN(BAABBA,0);                             EXPRD  302
   BETA=BAABBA; CALL CANONCL(BAABBA,MOD);                          EXPRD  303
   EX=LG('+',LG('*',I2,LG('|',N,I3)),                              EXPRD  304
      LG('+',LG('*',I24,LG('|',N,I4)),                             EXPRD  305
      LG('+',LG('*',I52,LG('|',N,I5)),                             EXPRD  306
         LG('*',I30,LG('|',N,I6)))));                              EXPRD  307
   L=LG(':',BAABBA,EX);                                            EXPRD  308
   IF PRNTI THEN CALL STRPN(L,0);                                  EXPRD  309
   BETA=L; CALL CANONCL(L,MOD);                                    EXPRD  310
   LA=LG('.',LA,L);                                                EXPRD  311
   IF PRNTI THEN CALL STRPN(LA,0);                                 EXPRD  312
   BETA=LA; CALL CANONCL(LA,MOD); END;                             EXPRD  313
                                                                   EXPRD  314
```

```
   IF 2*WA+4*WB < MOD THEN DO;                                          EXPRD  315
      BABBBA=LG(',',BABB,BA);                                           EXPRD  316
      IF PRNTI THEN CALL STRPN(BABBBA,0);                               EXPRD  317
      BETA=BABBBA; CALL CANONCL(BABBBA,MOD);                            EXPRD  318
      EX=LG('+',LG('*',I3,LG('|',N,I3)),                                EXPRD  319
         LG('+',LG('*',I27,LG('|',N,I4)),                               EXPRD  320
         LG('+',LG('*',I54,LG('|',N,I5)),                               EXPRD  321
                LG('*',I30,LG('|',N,I6)))));                            EXPRD  322
      L=LG(':',BABBBA,EX);                                              EXPRD  323
      IF PRNTI THEN CALL STRPN(L,0);                                    EXPRD  324
      BETA=L; CALL CANONCL(L,MOD);                                      EXPRD  325
      LA=LG('.',LA,L);                                                  EXPRD  326
      IF PRNTI THEN CALL STRPN(LA,0);                                   EXPRD  327
      BETA=LA; CALL CANONCL(LA,MOD); END;                               EXPRD  328
                                                                        EXPRD  329
   IF 3*WA+3*WB < MOD THEN DO;                                          EXPRD  330
      BABBAA=LG(',',BAB,BAA);                                           EXPRD  331
      IF PRNTI THEN CALL STRPN(BABBAA,0);                               EXPRD  332
      BETA=BABBAA; CALL CANONCL(BABBAA,MOD);                            EXPRD  333
      EX=LG('+',LG('*',I4,LG('|',N,I3)),                                EXPRD  334
         LG('+',LG('*',I21,LG('|',N,I4)),                               EXPRD  335
         LG('+',LG('*',I36,LG('|',N,I5)),                               EXPRD  336
                LG('*',I20,LG('|',N,I6)))));                            EXPRD  337
      L=LG(':',BABBAA,EX);                                              EXPRD  338
      IF PRNTI THEN CALL STRPN(L,0);                                    EXPRD  339
      BETA=L; CALL CANONCL(L,MOD);                                      EXPRD  340
      LA=LG('.',LA,L);                                                  EXPRD  341
      IF PRNTI THEN CALL STRPN(LA,0);                                   EXPRD  342
      BETA=LA; CALL CANONCL(LA,MOD); END;                               EXPRD  343
                                                                        EXPRD  344
   CAL=IRALST(INLSTR(LA,MTLIST(STR)));                                  EXPRD  345
   BETA=BETS;                                                           EXPRD  346
   IF PRNTI THEN PUT SKIP(2) LIST('EXIT EXPRD');                        EXPRD  347
   END EXPRD;                                                           EXPRD  348
```

```
GRPL:   PROC(STR)           BIT(1);                                          GRPL    0
        /* RETURNS '1'B IFF  STR IS A GROUP ELEMENT  */                      GRPL    1
        DCL (                                                               GRPL    2
             CH                                                             GRPL    3
            )     CHAR(1);                                                  GRPL    4
        DCL (                                                               GRPL    5
             STR                                                            GRPL    6
            )     FIXED BIN(31,0);                                          GRPL    7
        DCL (                                                               GRPL    8
             TOP                                                            GRPL    9
            )ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));               GRPL   10
        UNSPEC(CH)=SUBSTR(UNSPEC(TOP(STR)),9,8);                            GRPL   11
        RETURN(CH='G'|CH='U');                                              GRPL   12
        END GRPL;                                                           GRPL   13


IDNT:   PROC(STR)           BIT(1);                                          IDNT    0
        /* RETURNS '1'B IFF STR IS THE IDENTITY */                          IDNT    1
        DCL (                                                               IDNT    2
             STR                                                            IDNT    3
            )     FIXED BIN(31,0);                                          IDNT    4
        DCL (                                                               IDNT    5
             TOP                                                            IDNT    6
            )ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));               IDNT    7
        RETURN(SUBSTR(UNSPEC(TOP(STR)),9,8)=UNSPEC('U'));                   IDNT    8
        END IDNT;                                                           IDNT    9


INVEX:  PROC(STR)  FIXED BIN(31,0);                                         INVEX   0
```

```
        DCL (                                                                   INVEX    1
             STR,A,B,N                                                          INVEX    2
            )     FIXED BIN(31,0);                                              INVEX    3
        DCL (                                                                   INVEX    4
             BOT,CONT,TOP,MTLIST                                                INVEX    5
            )ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));                   INVEX    6
        DCL (                                                                   INVEX    7
             LG                                                                 INVEX    8
            )ENTRY(CHAR(1),FIXED BIN(31,0),FIXED BIN(31,0))                     INVEX    9
             RETURNS(FIXED BIN(31,0));                                          INVEX   10
        DCL (                                                                   INVEX   11
             MADNTP,INLSTR                                                      INVEX   12
            )ENTRY(FIXED BIN(31,0),FIXED BIN(31,0))                             INVEX   13
             RETURNS(FIXED BIN(31,0));                                          INVEX   14
        /* (A:N)' -> A:-N  */                                                   INVEX   15
        B=BOT(STR);                                                             INVEX   16
        A=CONT(MADNTP(B,2)+1);                                                  INVEX   17
        N=BOT(B);                                                               INVEX   18
        IF SUBSTR(UNSPEC(TOP(N)),9,24)=UNSPEC('00U')                            INVEX   19
           THEN A=LG(':',A,BOT(N));                                             INVEX   20
           ELSE A=LG(':',A,LG('U',N,0));                                        INVEX   21
        CALL IRALST(INLSTR(A,MTLIST(STR)));                                     INVEX   22
        END INVEX;                                                              INVEX   23


INVINV: PROC(STR);                                                              INVINV   0
        /* A'' -> A */                                                          INVINV   1
        DCL (                                                                   INVINV   2
             STR,A,CAL,LA,LR,I                                                  INVINV   3
            )     FIXED BIN(31,0);                                              INVINV   4
        DCL (                                                                   INVINV   5
             BOT,MTLIST,LNKR,CONT                                               INVINV   6
            )ENTRY(FIXED BIN(31,0))RETURNS(FIXED BIN(31,0));                    INVINV   7
```

```
        DCL (                                                                   INVINV   8
             INLSTR,NXTLFT,MADNTP                                               INVINV   9
            )ENTRY(FIXED BIN(31,0),FIXED BIN(31,0))                             INVINV  10
             RETURNS(FIXED BIN(31,0));                                          INVINV  11
        A=BOT(BOT(STR));                                                        INVINV  12
        CAL=MTLIST(STR); LR=LNKR(A); LA=LNKR(CONT(LR));                         INVINV  13
        DO I=1 TO 4;                                                            INVINV  14
          CAL=NXTLFT(CONT(MADNTP(A,I)+1),STR);                                  INVINV  15
          LA=LNKR(CONT(LA)); IF LA=LR THEN RETURN;                              INVINV  16
        END;                                                                    INVINV  17
        SIGNAL CONDITION(ALPHA);                                                INVINV  18
        END INVINV;                                                             INVINV  19


INVPRD: PROC(STR);                                                              INVPRD   0
        /* (AB)'-> B'A'    */                                                  INVPRD   1
        DCL (                                                                   INVPRD   2
        L                                                                       INVPRD   3
            )      FIXED BIN(31,0);                                             INVPRD   4
        DCL (                                                                   INVPRD   5
             A,B,STR,CAL,LA,LB                                                  INVPRD   6
            )      FIXED BIN(31,0);                                             INVPRD   7
        DCL (                                                                   INVPRD   8
             LG                                                                 INVPRD   9
            )ENTRY(CHAR(1),FIXED BIN(31,0),FIXED BIN(31,0))                     INVPRD  10
             RETURNS(FIXED BIN(31,0));                                          INVPRD  11
        DCL (                                                                   INVPRD  12
             BOT,CONT,MTLIST                                                    INVPRD  13
            )ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));                   INVPRD  14
        DCL (                                                                   INVPRD  15
             MADNTP,INLSTR                                                      INVPRD  16
            )ENTRY(FIXED BIN(31,0),FIXED BIN(31,0))                             INVPRD  17
             RETURNS(FIXED BIN(31,0));                                          INVPRD  18
```

```
        DCL (                                                          INVPRD  19
          SYMB                                                         INVPRD  20
         )ENTRY(FIXED BIN(31,0)) RETURNS(CHAR(1));                     INVPRD  21
        A=CONT(MADNTP(BOT(STR),2)+1);                                  INVPRD  22
        B=BOT(BOT(STR));                                               INVPRD  23
        IF SYMB(A)='''' THEN LA=CONT(MADNTP(A,2)+1);                   INVPRD  24
                        ELSE LA=LG('''',A,0);                          INVPRD  25
        IF SYMB(B)='''' THEN LB=CONT(MADNTP(B,2)+1);                   INVPRD  26
                        ELSE LB=LG('''',B,0);                          INVPRD  27
        A=LG('.',LB,LA);                                               INVPRD  28
        CALL IRALST(INLSTR(A,MTLIST(STR)));                            INVPRD  29
        END INVPRD;                                                    INVPRD  30


JACOBI: PROC(STR,MOD)    FIXED BIN(31,0);                              JACOBI   0
        /* JACOBI APPLIES THE JACOBI IDENTITY TO THE STRUCTURE STR */  JACOBI   1
        DCL (                                                          JACOBI   2
             A,B,C,L,V1,V2,V3,CAL,WA,WB,WC,BETS                        JACOBI   3
            )     FIXED BIN(31,0);                                     JACOBI   4
        DCL (AC,CB,BA,CA,BAC,CAB) FIXED BIN(31,0);                     JACOBI   5
        DCL (                                                          JACOBI   6
             STR,MOD                                                   JACOBI   7
            )     FIXED BIN(31,0);                                     JACOBI   8
        DCL (                                                          JACOBI   9
             CONT,BOT,WT,LIST,MTLIST                                   JACOBI  10
            )ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));          JACOBI  11
        DCL (                                                          JACOBI  12
             MADNTP,NXTLFT,INLSTR                                      JACOBI  13
            )ENTRY(FIXED BIN(31,0),FIXED BIN(31,0))                    JACOBI  14
             RETURNS(FIXED BIN(31,0));                                 JACOBI  15
        DCL (                                                          JACOBI  16
             LG                                                        JACOBI  17
            )ENTRY(CHAR(1),FIXED BIN(31,0),FIXED BIN(31,0))            JACOBI  18
```

```
     RETURNS(FIXED BIN(31,0));                                          JACOBI  19
DCL (STRPN,CANONCL) ENTRY (FIXED BIN(31,0),FIXED BIN(31,0));            JACOBI  20
DCL PRNTI BIT(1) EXT,BETA FIXED BIN(31,0)EXT;                           JACOBI  21
IF PRNTI THEN PUT SKIP(2) LIST('ENTER JACOBI');                         JACOBI  22
IF PRNTI THEN CALL STRPN(STR,0);                                        JACOBI  23
BETS=BETA;                                                              JACOBI  24
A=BOT(STR);                                                             JACOBI  25
L=CONT(MADNTP(STR,2)+1);                                                JACOBI  26
B=BOT(L);                                                               JACOBI  27
C=CONT(MADNTP(L,2)+1);                                                  JACOBI  28
AC=LG(',',A,C); BETA=AC; CALL CANONCL(AC,MOD);                          JACOBI  29
CB=LG(',',C,B); BETA=CB; CALL CANONCL(CB,MOD);                          JACOBI  30
BA=LG(',',B,A); BETA=BA; CALL CANONCL(BA,MOD);                          JACOBI  31
CA=LG(',',C,A); BETA=CA; CALL CANONCL(CA,MOD);                          JACOBI  32
BAC=LG(',',BA,C); BETA=BAC; CALL CANONCL(BAC,MOD);                      JACOBI  33
CAB=LG(',',CA,B); BETA=CAB; CALL CANONCL(CAB,MOD);                      JACOBI  34
WA=WT(A); WB=WT(B); WC=WT(C);                                           JACOBI  35
IF WA+WB+2*WC >= MOD THEN DO;                                           JACOBI  36
  L=LIST(9); CAL=NXTLFT(UNSPEC('0000'),L);  END;                        JACOBI  37
IF WA+WB +2*WC < MOD THEN L=LG(',',AC,CB);                              JACOBI  38
IF WA+2*WB+2*WC < MOD THEN L=LG('.',L,LG(',',CB,BAC));                  JACOBI  39
L=LG('.',L,LG(''',BAC,0));                                              JACOBI  40
IF WA+2*WB+WC < MOD THEN L=LG('.',L,LG(',',CB,BA));                     JACOBI  41
IF 2*WA+2*WB+WC<MOD THEN L=LG('.',L,LG(',',BA,LG(',',AC,B)));           JACOBI  42
IF 2*WA+WB+2*WC < MOD THEN                                              JACOBI  43
   L=LG('.',L,LG(',',AC,LG(''',LG(',',CA,B),0)));                       JACOBI  44
L=LG('.',L,CAB);                                                        JACOBI  45
IF 2*WA+WB+WC < MOD THEN L=LG('.',L,LG(',',BA,AC));                     JACOBI  46
IF 2*WA+WB+2*WC<MOD THEN L=LG('.',L,LG(',',AC,LG(',',CB,A)));           JACOBI  47
IF PRNTI THEN CALL STRPN(L,0);                                          JACOBI  48
CALL IRALST(INLSTR(L,MILIST(STR)));                                     JACOBI  49
IF PRNTI THEN PUT SKIP(2) LIST('EXIT JACOBI');                          JACOBI  50
BETA=BETS;                                                              JACOBI  51
END JACOBI;                                                             JACOBI  52
```

```
LEO:    PROC(STR,MOD);                                                  LEO      0
        /* LEFT EXPONENT OUT */                                         LEO      1
        /* N IS THE GREATEST INTEGER SUCH THAT                         LEO      2
                N*WT(B)+WT(A) <  MOD    */                            LEO      3
        DCL K FIXED BIN(31,0);                                          LEO      4
        DCL(                                                            LEO      5
            STR,MOD,A,B,X,BX,N,L,LA,LB,J,CAL,WTA,WTB,BETS,              LEO      6
            BAB,BABB,BA,LC,L1,L2,L3,L4                                  LEO      7
            )FIXED BIN(31,0);                                           LEO      8
        DCL(                                                            LEO      9
            WT,CONT,BOT,LIST,MILIST,TCP                                 LEO     10
           )ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));            LEO     11
        DCL(                                                            LEO     12
            NXTLFT,MADNTP,INLSTR                                        LEO     13
           )ENTRY(FIXED BIN(31,0),FIXED BIN(31,0))                      LEO     14
            RETURNS(FIXED BIN(31,0));                                   LEO     15
        DCL(                                                            LEO     16
            LG                                                          LEO     17
           )ENTRY(CHAR(1),FIXED BIN(31,0),FIXED BIN(31,0))              LEO     18
            RETURNS(FIXED BIN(31,0));                                   LEO     19
        DCL INT BIT(1);                                                 LEO     20
        DCL KC CHAR(10), (KT,NX) CHAR(2);                               LEO     21
        DCL PRNTI BIT(1) EXT;                                           LEO     22
        DCL(                                                            LEO     23
            STRPN                                                       LEO     24
          )ENTRY(FIXED BIN(31,0),FIXED BIN(31,0));                      LEO     25
        DCL(                                                            LEO     26
            CANONCL                                                     LEO     27
           )ENTRY(FIXED BIN(31,0),FIXED BIN(31,0));                     LEO     28
        DCL NCR ENTRY(CHAR(2),CHAR(2))RETURNS(FIXED BIN(31,0));         LEO     29
        DCL BETA FIXED BIN(31,0) EXT;                                   LEO     30
```

```
DCL SCNP ENTRY(FIXED BIN(31,0),FIXED BIN(31,0),                    LEO    31
                FIXED BIN(31,0),FIXED BIN(31,0));                  LEO    32
IF PRNTI THEN PUT SKIP(2) LIST('ENTER LED');                       LEO    33
IF PRNTI THEN CALL STRPN(STR,0);                                   LEO    34
BETS=BETA; BETA=L; INT='0'B;                                       LEO    35
BX=CONT(MADNTP(STR,2)+1); A=BOT(STR);                              LEO    36
B=CONT(MADNTP(BX,2)+1);   X=BOT(BX);                               LEO    37
LA=LIST(9); CAL=NXTLFT(UNSPEC('0000'),LA);                         LEO    38
LB=TOP(X);                                                         LEO    39
IF SUBSTR(UNSPEC(LB),9,8)=UNSPEC('I') THEN DO;                     LEO    40
   UNSPEC(NX)=SUBSTR(UNSPEC(LB),17,16); INT='1'B;                  LEO    41
   IF SUBSTR(NX,1,1)='0' THEN SUBSTR(NX,1,1)=' ';                  LEO    42
                                          END;                     LEO    43
IF INT THEN IF NX<' 1' THEN GOTO S2;                               LEO    44
WTA=WT(A); WTB=WT(B); IF WTA+WTB>= MOD THEN GOTO S2;               LEO    45
IF MOD>7 THEN SIGNAL CONDITION(ALPHA);                             LEO    46
CAL=IRALST(LA);                                                    LEO    47
L=LG(',',B,A);                                                     LEO    48
IF PRNTI THEN CALL STRPN(L,0);                                     LEO    49
BETA=L; CALL CANONCL(L,MOD);                                       LEO    50
BA,LA=L;                                                           LEO    51
IF INT THEN IF NX=' 1' THEN GOTO S2;                               LEO    52
LA=LG(':',L,X);                                                    LEO    53
IF PRNTI THEN CALL STRPN(LA,0);                                    LEO    54
BETA=LA; CALL CANONCL(LA,MOD);                                     LEO    55
N=(MOD-1-WTA)/WTB; KC=(9)'0' ||'1';                                LEO    56
BABB=0;                                                            LEO    57
DO J=2 TO N;                                                       LEO    58
   KC=KC+'1'; KT=SUBSTR(KC,8,2);                                   LEO    59
   IF INT THEN IF NX<KT THEN GOTO S1;                              LEO    60
   L=LG(',',L,B);                                                  LEO    61
   IF PRNTI THEN CALL STRPN(L,0);                                  LEO    62
   BETA=L; CALL CANONCL(L,MOD);                                    LEO    63
   IF J=2 THEN BAB=L;  IF J=3 THEN BABB=L;                         LEO    64
```

```
      IF INT THEN IF NX=KT THEN DO;                                          LEO    65
         LA=LG('.',LA,L);                                                    LEO    66
         IF PRNTI THEN CALL STRPN(LA,0);                                     LEO    67
         GOTO S1;                    END;                                    LEO    68
      IF INT THEN LB=NCR(NX,KT);                                             LEO    69
          ELSE DO; LB=LIST(9); CAL=NXTLFT(UNSPEC('01'||KT),LB);              LEO    70
                   LB=LG('|',X,LB);                                          LEO    71
              END;                                                           LEO    72
      LB=LG(':',L,LB);                                                       LEO    73
      IF PRNTI THEN CALL STRPN(LB,0);                                        LEO    74
      BETA=LB; CALL CANONCL(LB,MOD);                                         LEO    75
      LA=LG('.',LA,LB);                                                      LEO    76
   END;                                                                      LEO    77
                                                                             LEO    78
S1:   IF 2*WTA+3*WTB < MOD THEN DO;                                          LEO    79
      L=LG(',',BAB,BA);                                                      LEO    80
      BETA=L; CALL CANONCL(L,MOD);                                           LEO    81
      L1=LIST(9); CAL=NXTLFT(UNSPEC('0101'),L1);                             LEO    82
      L3=LIST(9); CAL=NXTLFT(UNSPEC('0103'),L3);                             LEO    83
      IF INT THEN DO;                                                        LEO    84
        KT=SUBSTR(CHAR(NX+'01'),8,2);                                        LEO    85
        LC=LG('+',NCR(NX,'03'),NCR(KT,'03'));                                LEO    86
        BETA=LC; CALL CANONCL(LC,MOD);                                       LEO    87
              END;                                                           LEO    88
         ELSE DO;                                                            LEO    89
        LC=LG('+',LG('|',X,L3),LG('|',LG('+',X,L1),L3));                     LEO    90
              END;                                                           LEO    91
      L=LG(':',L,LC);                                                        LEO    92
      BETA=L; CALL CANONCL(L,MOD);                                           LEO    93
      LA=LG('.',LA,L);                                                       LEO    94
                                  END;                                       LEO    95
   IF 2*WTA+4*WTB < MOD THEN DO;                                             LEO    96
      IF BABB=0 THEN GOTO S2;                                                LEO    97
      L=LG(',',BABB,BA);                                                     LEO    98
```

```
          BETA=L; CALL CANONCL(L,MOD);                                    LEO     99
          L2=LIST(9); CAL=NXTLFT(UNSPEC('0102'),L2);                      LEO    100
          L4=LIST(9); CAL=NXTLFT(UNSPEC('0104'),L4);                      LEO    101
          IF INT THEN DO;                                                 LEO    102
            LC=LG('*',L2,NCR(KT,'04'));                                   LEO    103
            BETA=LC; CALL CANONCL(LC,MOD);                                LEO    104
            LC=LG('+',NCR(NX,'04'),LC);                                   LEO    105
            BETA=LC; CALL CANONCL(LC,MOD);                                LEO    106
                END;                                                      LEO    107
              ELSE DO;                                                    LEO    108
            LC=LG('+',LG('|',X,L4),                                       LEO    109
                      LG('*',L2,LG('|',LG('+',X,L1),L4)));                LEO    110
                      END;                                                LEO    111
        L=LG(':',L,LC);                                                   LEO    112
        BETA=L; CALL CANONCL(L,MOD);                                      LEO    113
        LA=LG('.',LA,L);                                                  LEO    114
                                END;                                      LEO    115
                                                                          LEO    116
S2:     BETA=LA;                                                          LEO    117
        CALL SCNP(LA,LA,0,MOD);                                           LEO    118
        IF PRNTI THEN CALL STRPN(LA,0);                                   LEO    119
        CAL= IRALST(INLSTR(LA,MILIST(STR)));                              LEO    120
        BETA=BETS;                                                        LEO    121
        IF PRNTI THEN PUT SKIP(2) LIST('EXIT LEO');                       LEO    122
        END LEO;                                                          LEO    123


LG:     PROC(SYMB,LEFT,RIGHT)          FIXED BIN(31,0);                   LG       0
        /* LG GENERATES A LIST STRUCTURE OF STANDARD FORMAT */            LG       1
        DCL (                                                             LG       2
             LEFT,RIGHT                                                   LG       3
            )     FIXED BIN(31,0);                                        LG       4
        DCL SYMB CHAR(1);                                                 LG       5
```

```
        DCL (                                                                   LG        6
             A,CAL                                                              LG        7
            )      FIXED BIN(31,0);                                             LG        8
        DCL (                                                                   LG        9
             NXTLFT                                                             LG       10
            )ENTRY(FIXED BIN(31,0),FIXED BIN(31,0))                             LG       11
             RETURNS(FIXED BIN(31,0));                                          LG       12
        DCL (                                                                   LG       13
             LSSCPY,LIST                                                        LG       14
            )ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));                   LG       15
        A=LIST(9);                                                              LG       16
        IF SYMB='''|(SYMB='U' & RIGHT=0)                                        LG       17
          THEN CAL=NXTLFT(UNSPEC('000'||SYMB),NXTLFT(LEFT,A));                  LG       18
          ELSE CAL=NXTLFT(UNSPEC('000'||SYMB),                                  LG       19
                          NXTLFT(LEFT,NXTLFT(RIGHT,A)));                        LG       20
        RETURN(A);                                                              LG       21
        END LG;                                                                 LG       22


LINVOUT: PROC(STR,MOD);                                                         LINVOUT   0
        /* A',B ->(A,B)'(B,A,A')   */                                           LINVOUT   1
        DCL (                                                                   LINVOUT   2
             A,B,L,CAL,STR,MOD                                                  LINVOUT   3
            )      FIXED BIN(31,0);                                             LINVOUT   4
        DCL (                                                                   LINVOUT   5
             LG                                                                 LINVOUT   6
            )ENTRY(CHAR(1),FIXED BIN(31,0),FIXED BIN(31,0))                     LINVOUT   7
             RETURNS(FIXED BIN(31,0));                                          LINVOUT   8
        DCL (                                                                   LINVOUT   9
             CONT,BOT,WT,MILIST                                                 LINVOUT  10
            )ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));                   LINVOUT  11
        DCL (                                                                   LINVOUT  12
             MADNTP,INLSTR                                                      LINVOUT  13
```

```
      )ENTRY(FIXED BIN(31,0),FIXED BIN(31,0))                                  LINVOUT 14
       RETURNS(FIXED BIN(31,0));                                               LINVOUT 15
      A=BOT(CONT(MADNTP(STR,2)+1));                                            LINVOUT 16
      B=BOT(STR);                                                              LINVOUT 17
      L=LG('''',LG(',',A,B),0);                                                LINVOUT 18
      IF WT(B)+2*WT(A)<MOD THEN                                                LINVOUT 19
        L=LG('.',L,LG(',',LG(',',B,A),LG('''',A,0)));                          LINVOUT 20
      CALL IRALST(INLSTR(L,MTLIST(STR)));                                      LINVOUT 21
      END LINVOUT;                                                             LINVOUT 22


LPRDOUT: PROC(STR,MOD);                                                        LPRDOUT  0
      /* AB,C -> (A,C)(A,C,B)(B,C)  */                                         LPRDOUT  1
      DCL (                                                                    LPRDOUT  2
        C,L,B,A,CAL,STR,MOD,WA,WB,WC,BETS,LA                                   LPRDOUT  3
        )    FIXED BIN(31,0);                                                  LPRDOUT  4
      DCL (                                                                    LPRDOUT  5
        BOT,CONT,WT,LIST,MTLIST                                                LPRDOUT  6
        )ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));                      LPRDOUT  7
      DCL (                                                                    LPRDOUT  8
        MADNTP,NXTLFT,INLSTR                                                   LPRDOUT  9
        )ENTRY(FIXED BIN(31,0),FIXED BIN(31,0))                                LPRDOUT 10
        RETURNS(FIXED BIN(31,0));                                              LPRDOUT 11
      DCL (                                                                    LPRDOUT 12
        LG                                                                     LPRDOUT 13
        )ENTRY(CHAR(1),FIXED BIN(31,0),FIXED BIN(31,0))                        LPRDOUT 14
        RETURNS(FIXED BIN(31,0));                                              LPRDOUT 15
      DCL (STRPN,CANUNCL) ENTRY (FIXED BIN(31,0),FIXED BIN(31,0));            LPRDOUT 16
      DCL PRNTI BIT(1) EXT,BETA FIXED BIN(31,0)EXT;                            LPRDOUT 17
      IF PRNTI THEN PUT SKIP(2) LIST('ENTER LPRDOUT');                         LPRDOUT 18
      IF PRNTI THEN CALL STRPN(STR,0);                                         LPRDOUT 19
      BETS=BETA;                                                               LPRDOUT 20
      C=BOT(STR);                                                              LPRDOUT 21
```

```
        L=CONT(MADNTP(STR,2)+1);                                          LPRDOUT 22
        B=BUT(L);                                                         LPRDOUT 23
        A=CONT(MADNTP(L,2)+1);                                            LPRDOUT 24
        WA=WT(A); WB=WT(B); WC=WT(C);                                     LPRDOUT 25
        IF WA+WC>= MOD THEN DO;                                           LPRDOUT 26
          L=LIST(9); CAL=NXTLFT(UNSPEC('0000'),L);  END;                  LPRDOUT 27
        IF WA+WC < MOD THEN DO;                                           LPRDOUT 28
           L=LG(',',A,C);                                                 LPRDOUT 29
           IF PRNTI THEN CALL STRPN(L,0);                                 LPRDOUT 30
           BETA=L; CALL CANONCL(L,MOD);  END;                             LPRDOUT 31
        IF WA+WB+WC < MOD THEN DO;                                        LPRDOUT 32
           LA=LG(',',L,B);                                                LPRDOUT 33
           IF PRNTI THEN CALL STRPN(LA,0);                                LPRDOUT 34
           BETA=LA; CALL CANONCL(LA,MOD);                                 LPRDOUT 35
           L=LG('.',L,LA);  END;                                          LPRDOUT 36
        IF WB+WC < MOD THEN DO;                                           LPRDOUT 37
           LA=LG(',',B,C);                                                LPRDOUT 38
           IF PRNTI THEN CALL STRPN(LA,0);                                LPRDOUT 39
           BETA=LA; CALL CANONCL(LA,MOD);                                 LPRDOUT 40
           L=LG('.',L,LA);                                                LPRDOUT 41
           IF PRNTI THEN CALL STRPN(L,0);                                 LPRDOUT 42
           BETA=L; CALL CANONCL(L,MOD);  END;                             LPRDOUT 43
        CALL IRALST(INLSTR(L,MTLIST(STR)));                               LPRDOUT 44
        IF PRNTI THEN PUT SKIP(2) LIST('EXIT LPRDOUT');                   LPRDOUT 45
        BETA=BETS;                                                        LPRDOUT 46
        END LPRDOUT;                                                      LPRDOUT 47


LSTSTR: PROC(LST) FIXED BIN(31,0);                                        LSTSTR   0
        /* THIS PROCEDURE CONVERTS THE LINEAR LIST 'LST'                  LSTSTR   1
           TO A STANDARD BINARY STRUCTURE.                 */             LSTSTR   2
        DCL (                                                             LSTSTR   3
             LST,PH,P1,LA,P2,CAL                                          LSTSTR   4
```

```
        )     FIXED BIN(31,0);                                          LSTSTR   5
      DCL (                                                             LSTSTR   6
           LG                                                           LSTSTR   7
          )ENTRY(CHAR(1),FIXED BIN(31,0),FIXED BIN(31,0))               LSTSTR   8
           RETURNS(FIXED BIN(31,0));                                    LSTSTR   9
      DCL (                                                             LSTSTR  10
           LNKR,LIST,CONT,LCNTR                                         LSTSTR  11
          )ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));             LSTSTR  12
      DCL (                                                             LSTSTR  13
           NXTLFT,MADNBT                                                LSTSTR  14
          )ENTRY(FIXED BIN(31,0),FIXED BIN(31,0))                       LSTSTR  15
           RETURNS(FIXED BIN(31,0));                                    LSTSTR  16
      DCL SETIND ENTRY(FIXED BIN(31,0),FIXED BIN(31,0),                 LSTSTR  17
                       FIXED BIN(31,0),FIXED BIN(31,0));                LSTSTR  18
      PH,P1=LNKR(LST);                                                  LSTSTR  19
      P1=LNKR(CONT(P1));                                                LSTSTR  20
      IF P1=PH THEN DO;                                                 LSTSTR  21
        LA=LIST(9); CAL=NXTLFT(UNSPEC('0U00'),LA);                      LSTSTR  22
        RETURN(LA); END;                                                LSTSTR  23
      P2=LNKR(CONT(P1));                                                LSTSTR  24
      IF P2=PH THEN DO;                                                 LSTSTR  25
        LA=CONT(MADNBT(CONT(P1+1),2)+1);                                LSTSTR  26
        CALL SETIND(-1,-1,LCNTR(LA)+1,LA+1);                            LSTSTR  27
        RETURN(LA); END;                                                LSTSTR  28
      LA=LG('.',CONT(MADNBT(CONT(P1+1),2)+1),                           LSTSTR  29
                CONT(MADNBT(CONT(P2+1),2)+1));                          LSTSTR  30
L1:   P2=LNKR(CONT(P2));                                                LSTSTR  31
      IF P2=PH THEN RETURN(LA);                                         LSTSTR  32
      LA=LG('.',LA,CONT(MADNBT(CONT(P2+1),2)+1));                       LSTSTR  33
      GOTO L1;                                                          LSTSTR  34
      END LSTSTR;                                                       LSTSTR  35


LT:   PROC(A,B)          BIT(1)    RECURSIVE;                           LT       0
```

```
/* LT RETURNS '1'B IFF ALB */                                          LT      1
DCL (                                                                  LT      2
     WT,TOP,CONT,BOT                                                   LT      3
    )ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));                  LT      4
DCL (                                                                  LT      5
     WTA,WTB,A1,B1,A2,B2,A,B,LDA,LDB,TRA,TRB                           LT      6
    )     FIXED BIN(31,0);                                             LT      7
DCL (                                                                  LT      8
     MADNTP                                                            LT      9
    )ENTRY(FIXED BIN(31,0),FIXED BIN(31,0))                            LT     10
     RETURNS(FIXED BIN(31,0));                                         LT     11
DCL (                                                                  LT     12
     LT,EQ                                                             LT     13
    )ENTRY(FIXED BIN(31,0),FIXED BIN(31,0))                            LT     14
     RETURNS(BIT(1));                                                  LT     15
DCL SYMB ENTRY(FIXED BIN(31,0)) RETURNS(CHAR(1));                      LT     16
DCL (SA,SB) CHAR(1);                                                   LT     17
WTA=WT(A); WTB=WT(B); SA=SYMB(A); SB=SYMB(B);                          LT     18
IF WTA>WTB THEN RETURN('0'B);                                          LT     19
IF WTA<WTB THEN RETURN('1'B);                                          LT     20
IF SA=':' | SA='''' THEN RETURN(LT(CONT(MADNTP(A,2)+1),B));            LT     21
IF SB=':' | SB='''' THEN RETURN(LT(A,CONT(MADNTP(B,2)+1)));            LT     22
IF WTA=1 THEN RETURN(SA<SB);                                           LT     23
A1=CONT(MADNTP(A,2)+1); A2=BOT(A);                                     LT     24
B1=CONT(MADNTP(B,2)+1); B2=BOT(B);                                     LT     25
IF LT(A1,A2)                                                           LT     26
   THEN DO; LDA=A2; TRA=A1; END;                                       LT     27
   ELSE DO; LDA=A1; TRA=A2; END;                                       LT     28
IF LT(B1,B2)                                                           LT     29
   THEN DO; LDB=B2; TRB=B1; END;                                       LT     30
   ELSE DO; LDB=B1; TRB=B2; END;                                       LT     31
RETURN(LT(LDA,LDB)|                                                    LT     32
       (EQ(LDA,LDB)&LT(TRA,TRB))|                                      LT     33
       (EQ(LDA,LDB)&EQ(TRA,TRB)&LT(B1,A1)));                           LT     34
```

```
        END LT;                                                          LT      35

NCOM:   PROC(NGEN,W) FIXED BIN(31,0) RECURSIVE;                          NCOM     0
        /* THIS PROCEDURE CALCULATES AN UPPER BOUND TO THE               NCOM     1
           NUMBER OF BASIC COMMUTATORS FOR EACH WEIGHT UP TO             NCOM     2
           A GIVEN WEIGHT FOR A GIVEN NUMBER OF GENERATORS               NCOM     3
           NGEN.                                                         NCOM     4
           THE VALUES ARE ENTERED IN AN ARRAY NC          */             NCOM     5
        DCL NC(10) FIXED BIN(31,0) STATIC EXT;                           NCOM     6
        DCL (                                                            NCOM     7
             NGEN,W,NN,I                                                 NCOM     8
            )     FIXED BIN(31,0);                                       NCOM     9
        DCL (                                                            NCOM    10
             NCOM                                                        NCOM    11
             )ENTRY(FIXED BIN(31,0),FIXED BIN(31,0))                     NCOM    12
             RETURNS(FIXED BIN(31,0));                                   NCOM    13
        IF W>10 THEN SIGNAL CONDITION (ALPHA);                           NCOM    14
        IF W>10 THEN DO; PUT LIST('DUMMY STATEMENT');                    NCOM    15
                         SIGNAL CONDITION (ALPHA);  END;                 NCOM    16
        IF W=1 THEN DO; NC(1)=NGEN; RETURN(NGEN); END;                   NCOM    17
        NN=0;                                                            NCOM    18
        I=(W+1)/2; J=W-1;                                                NCOM    19
        DO I=I TO J;                                                     NCOM    20
           NN=NN+NCOM(NGEN,I)*NCOM(NGEN,W-I);                            NCOM    21
        END;                                                             NCOM    22
        NC(W)=NN; RETURN(NN);                                            NCOM    23
        END NCOM;                                                        NCOM    24


NCR:    PROC(N,R) FIXED BIN(31,0);                                       NCR      0
        /* NCR RETURNS A STRUCTURE CONTAINING THE INTEGER EQUQL          NCR      1
```

```
      TO THE NUMBER OF COMBINATIONS OF N THINGS R AT A TIME */          NCR      2
     DCL (L,I,CAL) FIXED BIN(31,0);                                     NCR      3
     DCL (N,R) CHAR(2);                                                 NCR      4
     DCL (DN,DR,PRD) FIXED DECIMAL;                                     NCR      5
     DCL(                                                               NCR      6
          LIST                                                          NCR      7
         )ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));              NCR      8
     DCL (                                                              NCR      9
          NXTLFT                                                        NCR     10
         )ENTRY(FIXED BIN(31,0),FIXED BIN(31,0))                        NCR     11
          RETURNS(FIXED BIN(31,0));                                     NCR     12
     IF N=0 THEN DO;                                                    NCR     13
        L=LIST (9); CAL=NXTLFT(UNSPEC('0100'),L);                       NCR     14
        RETURN(L);   END;                                               NCR     15
     IF R=0 THEN DO;                                                    NCR     16
        L=LIST(9); CAL=NXTLFT(UNSPEC('0100'),L);                        NCR     17
        RETURN(L);  END;                                                NCR     18
     DN=N; DR=R; PRD=1;                                                 NCR     19
     DO I=1 TO DR; PRD=PRD*(DN-I+1); END;                               NCR     20
     DO I=1 TO DR; PRD=PRD/(DR-I+1); END;                               NCR     21
     IF PRD>99 THEN SIGNAL CONDITION(ALPHA);                            NCR     22
     L=LIST(9);                                                         NCR     23
     CAL=NXTLFT(UNSPEC('0I' || SUBSTR(CHAR(PRD),7,2)),L);               NCR     24
     RETURN(L);                                                         NCR     25
     END NCR;                                                           NCR     26


NTERM:  PROC(STR)           BIT(1);                                     NTERM    0
     /* RETURNS'1'B IF STRUCTURE IS NON TERMINAL  */                    NTERM    1
     DCL (                                                              NTERM    2
          STR                                                           NTERM    3
         )     FIXED BIN(31,0);                                       NTERM    4
     DCL (                                                              NTERM    5
```

```
        LNKL,LNKR,CONT                                                  NTERM     6
           )ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));            NTERM     7
        RETURN(LNKL(CONT(STR))¬=LNKR(CONT(STR)));                       NTERM     8
        END NTERM;                                                      NTERM     9


NU:     PROC(STR) FLOAT BIN(53) RECURSIVE;                              NU        0
        DCL (                                                           NU        1
             WT,CONT,BOT                                                NU        2
           )ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));            NU        3
        DCL (                                                           NU        4
             MADNTP                                                     NU        5
           )ENTRY(FIXED BIN(31,0),FIXED BIN(31,0))                      NU        6
            RETURNS(FIXED BIN(31,0));                                   NU        7
        DCL (                                                           NU        8
             SYMB                                                       NU        9
           )ENTRY(FIXED BIN(31,0)) RETURNS(CHAR(1));                    NU       10
        DCL NN FLOAT BIN(53);                                           NU       11
        DCL (                                                           NU       12
             L,R,WL,WR,WS,I,STR                                         NU       13
           )     FIXED BIN(31,0);                                       NU       14
        DCL NU ENTRY(FIXED BIN(31,0)) RETURNS(FLOAT BIN(53));           NU       15
        DCL NC(10) FIXED BIN(31,0) STATIC EXT;                          NU       16
        DCL NA FIXED BIN(15,0);                                         NU       17
        DCL SYM CHAR(1);                                                NU       18
        DCL NTERM ENTRY(FIXED BIN(31,0)) RETURNS(BIT(1));               NU       19
        IF ¬NTERM(STR) THEN DO;                                         NU       20
           UNSPEC(NA)=(10)'0'B||SUBSTR(UNSPEC(SYMB(STR)),3,6); NN=NA;   NU       21
           RETURN(NN);    END;                                         NU       22
        L=CONT(MADNTP(STR,2)+1); R=BOT(STR);                            NU       23
        SYM=SYMB(STR);                                                  NU       24
        IF SYM=':' | SYM='''' THEN RETURN(NU(L));                       NU       25
        WS=WT(STR); WR=WT(R); WL=WS-1;                                  NU       26
```

```
      I=(WS+1)/2; NN=0;                                                    NU      27
      DO I=I TO WL;                                                        NU      28
         NN=NN+NC(I)*NC(WS-I)+NC(WR)*NU(L)+NU(R);                          NU      29
      END;                                                                 NU      30
      RETURN(NN);                                                          NU      31
      END NU;                                                              NU      32


ORDPROD: PROC(STR,MOD);                                                    ORDPROD  0
      DCL (                                                                ORDPROD  1
          P1,P2,P3,P4,PH,W1,W2,W3,K1,K2,K3,KA,KB,WA,WB,ORLA,ORLB,          ORDPROD  2
          WGT,ORDL,ORDR,ORL1,ORL2,ORL3,ORR1,ORR2,ORR3,ORRA,ORRB,           ORDPROD  3
          L,MOD,LST,CAL,STR,LA,HOLDLST,P5,LR                               ORDPROD  4
          )      FIXED BIN(31,0);                                          ORDPROD  5
      DCL (                                                                ORDPROD  6
          NEW                                                              ORDPROD  7
          )      BIT(1);                                                   ORDPROD  8
      DCL (                                                                ORDPROD  9
          LNKR,LNKL,CONT,DELETE,SUMEX,TOP,BOT,LIST,LSTSTR,MTLIST           ORDPROD 10
          )ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));                ORDPROD 11
      DCL (                                                                ORDPROD 12
          SUBST,MADNTP,MADNBT,NXTLFT,NXTRGT,INLSTR                         ORDPROD 13
          )ENTRY(FIXED BIN(31,0),FIXED BIN(31,0))                          ORDPROD 14
           RETURNS(FIXED BIN(31,0));                                       ORDPROD 15
      DCL (                                                                ORDPROD 16
           CANONCL,PROSPEC,STRPN                                           ORDPROD 17
         )ENTRY(FIXED BIN(31,0),FIXED BIN(31,0));                          ORDPROD 18
      DCL (                                                                ORDPROD 19
           LG                                                              ORDPROD 20
          )ENTRY(CHAR(1),FIXED BIN(31,0),FIXED BIN(31,0))                  ORDPROD 21
           RETURNS(FIXED BIN(31,0));                                       ORDPROD 22
      DCL PRNT FIXED BIN(31,0) EXT, PRNTI BIT(1) EXT;                      ORDPROD 23
      DCL BETA FIXED BIN(31,0) EXT,BETS FIXED BIN(31,0);                   ORDPROD 24
```

```
        DCL IDNT ENTRY(FIXED BIN(31,0))RETURNS(BIT(1));                    ORDPROD 25
        DCL F BIT(1);                                                      ORDPROD 26
        BETS=BETA;                                                         ORDPROD 27
        IF PRNTI THEN  PUT SKIP(2) LIST('ENTER ORDPROD');                  ORDPROD 28
        IF PRNTI THEN CALL STRPN(STR,0);                                   ORDPROD 29
        LST=LIST(9); CAL=PROSPEC(STR,LST);                                 ORDPROD 30
        PH,P1,P2,P3,P4=LNKR(LST);                                          ORDPROD 31
        WGT,ORDL,ORDR='0'B||(31)'1'B; NEW='1'B;                            ORDPROD 32
        F='0'B;                                                            ORDPROD 33
                                                                           ORDPROD 34
                                                                           ORDPROD 35
L1:     /* SCAN LIST FOR LEAST COMMUTATOR */                               ORDPROD 36
        /* P1 POINTS TO THE RIGHTMOST COLLECTED PART */                    ORDPROD 37
        /* P2 POINTS TO THE LEFTMOST &LEAST FACTOR                        ORDPROD 38
              CURRENTLY KNOWN DURING SCANNING  */                          ORDPROD 39
        /* P3 IS USED FOR SCANNING  */                                     ORDPROD 40
        /* PH MARKS THE HEADER  */                                         ORDPROD 41
                                                                           ORDPROD 42
        P3=LNKR(CONT(P3)); K3=CONT(P3+1);                                  ORDPROD 43
        IF P3=PH THEN GOTO L2;                                             ORDPROD 44
        W3=BOT(K3); ORL3=TOP(K3); ORR3=CONT(MADNTP(K3,2)+1);               ORDPROD 45
        IF W3 > WGT THEN GOTO L1;                                          ORDPROD 46
        IF W3 < WGT THEN GOTO L7;                                          ORDPROD 47
        IF ORL3 > ORDL THEN GOTO L1;                                       ORDPROD 48
        IF ORL3 < ORDL THEN GOTO L7;                                       ORDPROD 49
        IF ORR3 >= ORDR THEN GOTO L1;                                      ORDPROD 50
                                                                           ORDPROD 51
L7:     P2=P3; NEW='0'B; K2=K3;                                            ORDPROD 52
        WGT=W3; ORDL=ORL3; ORDR=ORR3; GOTO L1;                             ORDPROD 53
                                                                           ORDPROD 54
L2:     /* THIS SECTION MOVES THE LEAST COMMUTATOR FOUND                  ORDPROD 55
             IN THE LAST SECTION     */                                    ORDPROD 56
        IF NEW='1'B THEN GOTO L3;                                          ORDPROD 57
        IF P2=LNKR(CONT(PH)) THEN DO; P1=P2; GOTO L3;  END;                ORDPROD 58
```

```
      IF P2=LNKR(CONT(P1)) THEN DO; P1=P2; GOTO L3; END;                    ORDPROD 59
      IF P3=LNKR(CONT(P1)) THEN DO;                                         ORDPROD 60
         HOLDLST=LIST(9); CAL=NXTLFT(CONT(P1+1),HOLDLST);                   ORDPROD 61
         GOTO L4;          END;                                             ORDPROD 62
      IF P1=PH THEN DO;                                                     ORDPROD 63
         CAL=NXTRGT(K2,P1); P1=LNKR(CONT(P1)); P2=LNKR(CONT(P2));           ORDPROD 64
         HOLDLST=LIST(9); CAL=NXTLFT(CONT(P1+1),HOLDLST);                   ORDPROD 65
         CAL=DELETE(LNKL(CONT(P2))); P3=P1; GOTO L4;  END;                  ORDPROD 66
      K1=CONT(P1+1); W1=BOT(K1); ORL1=TOP(K1);                              ORDPROD 67
      ORR1=CONT(MADNTP(K1,2)+1);                                            ORDPROD 68
      HOLDLST=LIST(9); CAL=NXTLFT(K2,HOLDLST);                              ORDPROD 69
      CAL=NXTRGT(K2,P1); P1=LNKR(CONT(P1));                                 ORDPROD 70
      P3=P1; P2=LNKR(CONT(P2)); CAL=DELETE(LNKL(CONT(P2)));                 ORDPROD 71
                                                                            ORDPROD 72
                                                                            ORDPROD 73
      /* SCAN REGION BETWEEN P1 & P2 TO INTRODUCE EXTRA                    ORDPROD 74
        TERMS IF NECESSARY  */                                              ORDPROD 75
                                                                            ORDPROD 76
L4:   P3=LNKR(CONT(P3)); K3=CONT(P3+1);                                     ORDPROD 77
      IF P3=PH THEN DO; CAL=IRALST(HOLDLST); GOTO L3; END;                  ORDPROD 78
                                                                            ORDPROD 79
      IF P3=P2 THEN DO;                                                     ORDPROD 80
         IF PRNTI & F THEN DO;                                              ORDPROD 81
            F='0'B; L=LSTSTR(LST);                                          ORDPROD 82
            CALL STRPN(L,0); CAL=IRALST(L);                                 ORDPROD 83
                             END;                                           ORDPROD 84
         P3=LNKL(CONT(P3));                                                 ORDPROD 85
         NEW='1'B; CAL=IRALST(HOLDLST);                                     ORDPROD 86
L8:      P3=LNKR(CONT(P3)); K3=CONT(P3+1);                                  ORDPROD 87
         IF P3=PH THEN GOTO L2;                                             ORDPROD 88
         W3=BOT(K3); ORL3=TOP(K3); ORR3=CONT(MADNTP(K3,2)+1);               ORDPROD 89
         IF W3>WGT THEN GOTO L8;                                            ORDPROD 90
         IF ORL3>ORDL THEN GOTO L8;                                         ORDPROD 91
         IF ORR3>ORDR THEN GOTO L8;                                         ORDPROD 92
```

```
          P2=P3; NEW='0'B; K2=K3;                                          ORDPROD 93
          P3=LNKR(CONT(P3)); K3=CONT(P3+1);                                ORDPROD 94
          GOTO L2;   END;                                                  ORDPROD 95
                                                                           ORDPROD 96
       W3=BOT(K3); ORL3=TOP(K3);                                           ORDPROD 97
       ORR3=CONT(MADNTP(K3,2)+1);                                          ORDPROD 98
       K1=BOT(HOLDLST); W1=BOT(K1);                                        ORDPROD 99
       IF W3+W1 < MOD THEN DO;                                             ORDPROD100
          L=LG(',',CONT(MADNBT(K3,2)+1),CONT(MADNBT(K1,2)+1));             ORDPROD101
       BETA=L; CALL CANONCL(L,MOD);                                        ORDPROD102
       P5=LNKR(CONT(P3));                                                  ORDPROD103
       LA=LIST(9); CAL=PROSPEC(L,LA); CAL=INLSTR(LA,P3);                   ORDPROD104
       P3=LNKL(CONT(P5));                                                  ORDPROD105
       F='1'B;                                                             ORDPROD106
                                                 END;                      ORDPROD107
       GOTO L4;                                                            ORDPROD108
                                                                           ORDPROD109
L3:    IF P4=PH THEN  P4=LNKR(CONT(P4));                                   ORDPROD110
       P3=P4; KA=CONT(P3+1); WA=BOT(KA);                                   ORDPROD111
       ORLA=TOP(KA); ORRA=CONT(MADNTP(KA,2)+1);                            ORDPROD112
L5:    P3=LNKR(CONT(P3)); IF P3=PH THEN GOTO L6;                           ORDPROD113
       KB=CONT(P3+1); WB=BOT(KB);                                          ORDPROD114
       ORLB=TOP(KB); ORRB=CONT(MADNTP(KB,2)+1);                            ORDPROD115
       IF WA¬=WB THEN GOTO J2;                                             ORDPROD116
       IF ORLA¬=ORLB THEN GOTO J2;                                         ORDPROD117
       IF ORRA¬=ORRB THEN GOTO J2;                                         ORDPROD118
        F='1'B;                                                            ORDPROD119
       L=LG('.',CONT(MADNBT(KA,2)+1),CONT(MADNBT(KB,2)+1));               ORDPROD120
       CAL=SUMEX(L);                                                       ORDPROD121
       BETA=BOT(L); CALL CANONCL(BOT(L),MOD);                              ORDPROD122
       BETA=L; CALL CANONCL(L,MOD);                                        ORDPROD123
       LA=LIST(9); CAL=PROSPEC(L,LA); CAL=INLSTR(LA,P3);                   ORDPROD124
       IF P1=LNKL(CONT(P3)) THEN P1=P3;                                    ORDPROD125
       CAL=DELETE(LNKL(CONT(P3)));                                         ORDPROD126
```

```
          P3=LNKR(CONT(P3));                                              ORDPROD127
          IF P1=LNKL(CONT(P3)) THEN P1=P3;                                ORDPROD128
          CAL=DELETE(LNKL(CONT(P3))); KA=CONT(P3+1);                      ORDPROD129
          GOTO L5;                                                        ORDPROD130
 J2:      /* SUMMATION NOT POSSIBLE */                                    ORDPROD131
          WA=WB; ORLA=ORLB; ORRA=ORRB; KA=KB;                             ORDPROD132
          GOTO L5;                                                        ORDPROD133
                                                                          ORDPROD134
 L6:      ORDL,ORDR,WGT='0'B||(31)'1'B; NEW='1'B;                         ORDPROD135
          IF PRNTI & F THEN DO;                                           ORDPROD136
             F='0'B; L=LSTSTR(LST);                                       ORDPROD137
             CALL STRPN(L,0); CAL=IRALST(L);                              ORDPROD138
                           END;                                           ORDPROD139
          P2,P3,P4=P1;                                                    ORDPROD140
          IF LNKR(CONT(P1))¬= PH THEN GOTO L1;                            ORDPROD141
                                                                          ORDPROD142
                                                                          ORDPROD143
          L=LSTSTR(LST); CAL=IRALST(LST);                                 ORDPROD144
          CAL =IRALST(INLSTR(L,MTLIST(STR)));                             ORDPROD145
          IF PRNTI THEN CALL STRPN(STR,0);                                ORDPROD146
          BETA=BETS;                                                      ORDPROD147
          IF PRNTI THEN PUT SKIP(2) LIST('EXIT ORDPROD');                 ORDPROD148
          END ORDPROD;                                                    ORDPROD149


PERMCOM:  PROC(STR);                                                      PERMCOM   0
          /* A,B -> (B,A)' */                                             PERMCOM   1
          DCL (                                                           PERMCOM   2
               L,A,B,STR,CAL                                              PERMCOM   3
              )      FIXED BIN(31,0);                                     PERMCOM   4
          DCL (                                                           PERMCOM   5
               CONT,BOT,MTLIST                                            PERMCOM   6
              )ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));           PERMCOM   7
```

```
        DCL (                                                                   PERMCOM  8
         MADNTP,INLSTR                                                          PERMCOM  9
        )ENTRY(FIXED BIN(31,0),FIXED BIN(31,0))                                 PERMCOM 10
         RETURNS(FIXED BIN(31,0));                                              PERMCOM 11
        DCL (                                                                   PERMCOM 12
         LG                                                                     PERMCOM 13
        )ENTRY(CHAR(1),FIXED BIN(31,0),FIXED BIN(31,0))                         PERMCOM 14
         RETURNS(FIXED BIN(31,0));                                              PERMCOM 15
        DCL IRALST ENTRY(FIXED BIN(31,0));                                      PERMCOM 16
        A=CONT(MADNTP(STR,2)+1);                                                PERMCOM 17
        B=BOT(STR);                                                             PERMCOM 18
        A=  (LG('''',LG(',',B,A),0));                                           PERMCOM 19
        CALL IRALST(INLSTR(A,MTLIST(STR)));                                     PERMCOM 20
        END PERMCOM;                                                            PERMCOM 21


PREC:    PROC(STR) FIXED BIN(31,0);                                             PREC     0
/* THE PARAM STR IS A STRUCTURE.                                               PREC     1
THE PROC RETURNS A VALUE WHICH INDICATES THE ORDER OF                           PREC     2
PRECEDENCE OF AN OPERATOR.  THE LARGER THE VALUE                                PREC     3
THE STRONGER THE OPERATOR BOND.                                                 PREC     4
IT RETURNS  THE VALUE  20 IF THE STR IS TERMINAL    */                          PREC     5
        DCL (STR,I) FIXED BIN(31,0);                                            PREC     6
        DCL S CHAR(1);                                                          PREC     7
         DCL (                                                                  PREC     8
              SYMB                                                              PREC     9
             )ENTRY(FIXED BIN(31,0)) RETURNS(CHAR(1));                          PREC    10
         DCL (                                                                  PREC    11
              NTERM                                                             PREC    12
             )ENTRY(FIXED BIN(31,0)) RETURNS(BIT(1));                           PREC    13
        DCL CH(10) CHAR(1) STATIC                                               PREC    14
                 INITIAL('U','-','+','*','|','/',',',')','.','''',':');        PREC    15
         DCL PR(10) FIXED BIN(31,0) STATIC                                      PREC    16
```

```
        INITIAL(10,1,1,2,3,4,5,6,10,7);                                  PREC    17
      IF ¬NTERM(STR) THEN RETURN(20);                                    PREC    18
      S=SYMB(STR);                                                       PREC    19
       DO I=1 TO 10; IF CH(I)=S THEN RETURN(PR(I)); END;                 PREC    20
      PUT LIST('INVALID OPERATOR'); RETURN(20);                          PREC    21
      END PREC;                                                          PREC    22


PROD:  PROC(A,B) FIXED BIN(31,0);                                        PROD     0
       /* ADDS TWO INTEGERS IN LIST FORMAT AND RETURNS THE SUM          PROD     1
        IN LIST FORMAT */                                                PROD     2
       DCL (AM,BM) BIT(1);                                               PROD     3
       DCL (DL,DR) FIXED DEC(5,0);                                       PROD     4
       DCL (A,B,L,R,CAL) FIXED BIN(31,0);                                PROD     5
       DCL CT CHAR(2);                                                   PROD     6
       DCL(                                                              PROD     7
           LIST,BOI                                                      PROD     8
          )ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));              PROD     9
       DCL (                                                             PROD    10
           SYMB                                                          PROD    11
          )ENTRY(FIXED BIN(31,0)) RETURNS(CHAR(1));                      PROD    12
       DCL (                                                             PROD    13
           NTERM                                                         PROD    14
          )ENTRY(FIXED BIN(31,0)) RETURNS(BIT(1));                       PROD    15
       DCL (                                                             PROD    16
           NXTLFT                                                        PROD    17
          )ENTRY(FIXED BIN(31,0),FIXED BIN(31,0))                        PROD    18
           RETURNS(FIXED BIN(31,0));                                     PROD    19
       DCL (                                                             PROD    20
           LG                                                            PROD    21
          )ENTRY(CHAR(1),FIXED BIN(31,0),FIXED BIN(31,0))                PROD    22
           RETURNS(FIXED BIN(31,0));                                     PROD    23
      AM='0'B; BM='0'B;                                                  PROD    24
```

```
      IF NTERM(A) THEN IF SYMB(A)='U' THEN                                  PROD  25
                       DO; AM='1'B; L=BOT(BOT(A)); END;                     PROD  26
                                   ELSE SIGNAL CONDITION(ALPHA);            PROD  27
                  ELSE L=BOT(A);                                            PROD  28
      IF NTERM(B) THEN IF SYMB(B)='U' THEN                                  PROD  29
                       DO; BM='1'B; R=BOT(BOT(B)); END;                     PROD  30
                                    ELSE SIGNAL CONDITION(ALPHA);           PROD  31
                  ELSE R=BOT(B);                                            PROD  32
      UNSPEC(CT)=SUBSTR(UNSPEC(L),17,16);                                   PROD  33
      DL=CT; IF AM THEN DL=-DL;                                             PROD  34
      UNSPEC(CT)=SUBSTR(UNSPEC(R),17,16);                                   PROD  35
      DR=CT; IF BM THEN DR=-DR;                                             PROD  36
      DL=DL*DR;                                                             PROD  37
      IF DL>99 THEN SIGNAL CONDITION(ALPHA);                                PROD  38
      IF DL=0 THEN DO;                                                      PROD  39
         R=LIST(9); CAL=NXTLFT(UNSPEC('0I00'),R);                           PROD  40
        RETURN(R); END;                                                     PROD  41
      IF DL<0 THEN DO;                                                      PROD  42
         DL=-DL;                                                            PROD  43
        R=LIST(9); CAL=NXTLFT(UNSPEC('0I'||SUBSTR(CHAR(DL),7,2)),R);        PROD  44
         R=LG('U',R,0);                                                     PROD  45
         RETURN(R); END;                                                    PROD  46
      IF DL>0 THEN DO;                                                      PROD  47
        R=LIST(9); CAL=NXTLFT(UNSPEC('0I'||SUBSTR(CHAR(DL),7,2)),R);        PROD  48
         RETURN(R);                                                         PROD  49
      SIGNAL CONDITION (ALPHA);                                             PROD  50
      END PROD;                                                             PROD  51


PRODEXP: PROC(STR)  FIXED BIN(31,0);                                        PRODEXP  0
      /* (A:M):N -> A:(MN)   */                                             PRODEXP  1
      /* UNARY MINUS IF PRESENT IS RAISED TO THE TOP OF                     PRODEXP  2
        THE STRUCTURE     */                                               PRODEXP  3
```

```
/* STR IS A STRUCTURE WITH OPERATOR '**'. IT IS ASSUMED THAT THE
LEFT AND RIGHT SUBSTRUCTURE FORM AND IT
CONVERTS THE STRUCTURE                                          */
      DCL (
       STR,A,B,M,N,L,CAL
      ) FIXED BIN(31,0);                                                PRODEXP  4
      DCL (                                                             PRODEXP  5
       TOP,BOT,CONT,MTLIST                                              PRODEXP  6
      )ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));                 PRODEXP  7
      DCL (                                                             PRODEXP  8
       LG                                                               PRODEXP  9
      )ENTRY(CHAR(1),FIXED BIN(31,0),FIXED BIN(31,0))                   PRODEXP 10
       RETURNS(FIXED BIN(31,0));                                        PRODEXP 11
      DCL (                                                             PRODEXP 12
       MADNTP,INLSTR                                                    PRODEXP 13
      )ENTRY(FIXED BIN(31,0),FIXED BIN(31,0))                           PRODEXP 14
       RETURNS(FIXED BIN(31,0));                                        PRODEXP 15
      DCL (                                                             PRODEXP 16
       LU,RU                                                            PRODEXP 17
      )     BIT(1);                                                     PRODEXP 18
      DCL IRALST ENTRY(FIXED BIN(31,0));                                PRODEXP 19
      L=CONT(MADNTP(STR,2)+1);                                          PRODEXP 20
      A=CONT(MADNTP(L,2)+1);                                            PRODEXP 21
      M=BOT(L);                                                         PRODEXP 22
      N=BOT(STR);                                                       PRODEXP 23
      LU='0'B;  RU='0'B;                                                PRODEXP 24
      IF SUBSTR(UNSPEC(TOP(M)),9,24)=UNSPEC('00U') THEN LU='1'B;        PRODEXP 25
      IF SUBSTR(UNSPEC(TOP(N)),9,24)=UNSPEC('00U') THEN RU='1'B;        PRODEXP 26
      IF ¬LU & ¬RU THEN B=LG('*',M,N);                                  PRODEXP 27
      IF LU & RU THEN B=LG('*',BOT(M),BOT(N));                          PRODEXP 28
      IF LU & ¬RU THEN B=LG('U',LG('*',BOT(M),N),0);                    PRODEXP 29
      IF ¬LU &RU THEN B=LG('U',LG('*',M,BOT(N)),0);                     PRODEXP 30
      CAL= IRALST(INLSTR(LG(':',A,B),MTLIST(STR)));                     PRODEXP 31
      END PRODEXP;                                                      PRODEXP 32
                                                                        PRODEXP 33
                                                                        PRODEXP 34

PRODINT: PROC(STR);                                                     PRODINT  0
```

```
/* STR IS A STRUCTURE WITH OPERATOR '*'. IT IS ASSUMED THAT THE                PRODINT  1
LEFT AND RIGHT PARTS ARE IN CANONIC FORM AND IT                                 PRODINT  2
CONVERTS THE STRUCTURE TO CANONIC FORM.                              */         PRODINT  3
        DCL(                                                                    PRODINT  4
             L,R,BR,CAL,STR                                                     PRODINT  5
             )FIXED BIN(31,0);                                                  PRODINT  6
        DCL(                                                                    PRODINT  7
             TL,TR                                                              PRODINT  8
        )CHAR(4);                                                               PRODINT  9
        DCL(                                                                    PRODINT 10
             SUBST,MADNTP,INLSTR,PROD,SUBSTP,NXTLFT                             PRODINT 11
            )ENTRY(FIXED BIN(31,0),FIXED BIN(31,0))                             PRODINT 12
             RETURNS(FIXED BIN(31,0));                                          PRODINT 13
        DCL(                                                                    PRODINT 14
             LG                                                                 PRODINT 15
            )ENTRY(CHAR(1),FIXED BIN(31,0),FIXED BIN(31,0))                     PRODINT 16
             RETURNS(FIXED BIN(31,0));                                          PRODINT 17
        DCL(                                                                    PRODINT 18
             SYMB                                                               PRODINT 19
            )ENTRY(FIXED BIN(31,0)) RETURNS(CHAR(1));                           PRODINT 20
        DCL(                                                                    PRODINT 21
             CONT,MTLIST,TOP,BOT,LIST                                           PRODINT 22
            )ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));                   PRODINT 23
        DCL (                                                                   PRODINT 24
            NTERM                                                               PRODINT 25
            )ENTRY(FIXED BIN(31,0)) RETURNS(BIT(1));                            PRODINT 26
        DCL(WL,WR) CHAR(1);                                                     PRODINT 27
        DCL (LT,RT) CHAR(3);                                                    PRODINT 28
        DCL (WRA,WRB) CHAR(1);                                                  PRODINT 29
        L=CONT(MADNTP(STR,2)+1); R=BOT(STR);                                    PRODINT 30
        UNSPEC(TL)=UNSPEC(TOP(L)); UNSPEC(TR)=UNSPEC(TOP(R));                   PRODINT 31
        WL=SUBSTR(TL,2,1); WR=SUBSTR(TR,2,1);                                   PRODINT 32
        LT=SUBSTR(TL,2,3); RT=SUBSTR(TR,2,3);                                   PRODINT 33
        IF WL='I' & WR='I' THEN                                                 PRODINT 34
```

```
      DO; L=PROD(L,R); GOTO RET; END;                                          PRODINT 35
   IF LT='OOU' & WR='I' THEN DO;                                               PRODINT 36
      UNSPEC(WRA)=SUBSTR(UNSPEC(TOP(BOT(L))),9,8);                             PRODINT 37
      IF WRA='I' THEN DO; L=PROD(L,R); GOTO RET; END;                          PRODINT 38
                        END;                                                   PRODINT 39
   IF WL='I' & RT='OOU' THEN DO;                                               PRODINT 40
      UNSPEC(WRB)=SUBSTR(UNSPEC(TOP(BOT(R))),9,8);                             PRODINT 41
      IF WRB='I' THEN DO; L=PROD(L,R); GOTO RET; END;                          PRODINT 42
                        END;                                                   PRODINT 43
   IF LT='OOU' & RT='OOU' THEN DO;                                             PRODINT 44
      UNSPEC(WRA)=SUBSTR(UNSPEC(TOP(BOT(L))),9,8);                             PRODINT 45
      UNSPEC(WRB)=SUBSTR(UNSPEC(TOP(BOT(R))),9,8);                             PRODINT 46
      IF WRA='I' &WRB='I' THEN DO;                                             PRODINT 47
        L=PROD(L,R); GOTO RET; END;                                            PRODINT 48
                          END;                                                 PRODINT 49
   IF ¬NTERM(L) & ¬NTERM(R) THEN RETURN;                                       PRODINT 50
   IF LT='IOO' | LT='I O' | RT='IOO' | RT='I O' THEN DO;                       PRODINT 51
      L=LIST(9); CAL=NXTLFT(UNSPEC('OIOO'),L); GOTO RET; END;                  PRODINT 52
   IF SUBSTR(TR,2,1)='I' THEN DO;  L=LG('*',R,L); GOTO RET; END;               PRODINT 53
   IF SUBSTR(TR,2,3)='OOU' THEN DO;                                            PRODINT 54
/C.SYSIN DD *                                                                  PRODINT 55
   END PRODINT;                                                                PRODINT 56
```

```
PROSPEC: PROC(STR,LST) RECURSIVE;                                              PROSPEC  0
        /* THIS PROCEDURE TRANSFORMS THEN BINARY STRUCTURE STR                 PROSPEC  1
          INTO A LINEAR LIST OF LISTS.  THE LATTER IS ATTACHED                 PROSPEC  2
          TO THE EMPTY LIST LST.  */                                           PROSPEC  3
         DCL (                                                                 PROSPEC  4
           CAL,NUL,NUR,STR,LST,LA,L,R                                          PROSPEC  5
            ) FIXED BIN(31,0);                                                 PROSPEC  6
         DCL NUA FLOAT BIN(53);                                                PROSPEC  7
         DCL (                                                                 PROSPEC  8
```

```
      SYMB
      )ENTRY(FIXED BIN(31,0)) RETURNS(CHAR(1));                          PROSPEC  9
    DCL (                                                                PROSPEC 10
      NXTLFT,MADNTP,PROSPEC                                              PROSPEC 11
      )ENTRY(FIXED BIN(31,0),FIXED BIN(31,0))                            PROSPEC 12
       RETURNS(FIXED BIN(31,0));                                         PROSPEC 13
    DCL (                                                                PROSPEC 14
       CONT,BOT,LIST,WT                                                  PROSPEC 15
      )ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));                  PROSPEC 16
    DCL NU ENTRY(FIXED BIN(31,0)) RETURNS(FLOAT BIN(53));                PROSPEC 17
    DCL IDNT ENTRY(FIXED BIN(31,0)) RETURNS(BIT(1));                     PROSPEC 18
    IF IDNT(STR) THEN RETURN;                                            PROSPEC 19
    IF SYMB(STR)¬='.' THEN DO;                                           PROSPEC 20
      LA=LIST(9); NUA=NU(STR);                                           PROSPEC 21
    NUL=NUA/ 1E31B; NUR=NUA-NUL* 1E31B;                                  PROSPEC 22
      CAL=NXTLFT(NUL,LA); CAL=NXTLFT(NUR,LA);                            PROSPEC 23
      CAL=NXTLFT(STR,LA); CAL=NXTLFT(WT(STR),LA);                        PROSPEC 24
       CAL=NXTLFT(LA,LST); RETURN;                                       PROSPEC 25
                          END;                                           PROSPEC 26
  L=CONT(MADNTP(STR,2)+1); R=BOT(STR);                                   PROSPEC 27
   CALL PROSPEC(L,LST); CALL PROSPEC(R,LST);                             PROSPEC 28
  END PROSPEC;                                                           PROSPEC 29
                                                                         PROSPEC 30


REO:  PROC(STR,MOD);                                                     REO      0
      /* RIGHT EXPONENT OUT */                                           REO      1
      /* N IS THE GREATEST INTEGER SUCH THAT                            REO      2
              WT(A)+N.WT(B) < MOD                    */                  REO      3
      DCL K FIXED BIN(31,0);                                             REO      4
      DCL(                                                               REO      5
          STR,MOD,A,B,X,BX,N,L,LA,LB,J,CAL,WTA,WTB,BETS,                 REO      6
          BAB,BABB,BA,LC,L1,L2,L3,L4                                     REO      7
          )FIXED BIN(31,0);                                              REO      8
```

```
DCL(                                                                    REO       9
  WT,CONT,BOT,LIST,MTLIST,TOP                                           REO      10
  )ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));                     REO      11
DCL(                                                                    REO      12
  NXTLFT,MADNTP,INLSTR                                                  REO      13
  )ENTRY(FIXED BIN(31,0),FIXED BIN(31,0))                               REO      14
  RETURNS(FIXED BIN(31,0));                                             REO      15
DCL(                                                                    REO      16
  LG                                                                    REO      17
  )ENTRY(CHAR(1),FIXED BIN(31,0),FIXED BIN(31,0))                       REO      18
  RETURNS(FIXED BIN(31,0));                                             REO      19
DCL INT BIT(1);                                                         REO      20
DCL KC CHAR(10), (KT,NX) CHAR(2);                                       REO      21
DCL PRNTI BIT(1) EXT;                                                   REO      22
DCL(                                                                    REO      23
  STRPN                                                                 REO      24
  )ENTRY(FIXED BIN(31,0),FIXED BIN(31,0));                              REO      25
DCL(                                                                    REO      26
  CANONCL                                                               REO      27
  )ENTRY(FIXED BIN(31,0),FIXED BIN(31,0));                              REO      28
DCL BETA FIXED BIN(31,0) EXT;                                           REO      29
DCL SCNP ENTRY(FIXED BIN(31,0),FIXED BIN(31,0),                         REO      30
         FIXED BIN(31,0),FIXED BIN(31,0));                              REO      31
DCL NCR ENTRY(CHAR(2),CHAR(2))RETURNS(FIXED BIN(31,0));                 REO      32
IF PRNTI THEN PUT SKIP(2) LIST('ENTER REO');                            REO      33
IF PRNTI THEN CALL STRPN(STR,0);                                        REO      34
BETS=BETA; BETA=L; INT='0'B;                                            REO      35
A=CONT(MADNTP(STR,2)+1); BX=BOT(STR);                                   REO      36
B=CONT(MADNTP(BX,2)+1);   X=BOT(BX);                                    REO      37
LA=LIST(9); CAL=NXTLFT(UNSPEC('0000'),LA);                              REO      38
LB=TOP(X);                                                              REO      39
IF SUBSTR(UNSPEC(LB),9,8)=UNSPEC('I') THEN DO;                          REO      40
   UNSPEC(NX)=SUBSTR(UNSPEC(LB),17,16); INT='1'B;                       REO      41
   IF SUBSTR(NX,1,1)='0' THEN SUBSTR(NX,1,1)=' ';                       REO      42
```

```
                                                    END;                                    REO     43
IF INT THEN IF NX<'1' THEN GOTO S2;                                                         REO     44
WTA=WT(A); WTB=WT(B); IF WTA+WTB>= MOD THEN GOTO S2;                                        REO     45
IF MOD>7 THEN SIGNAL CONDITION(ALPHA);                                                      REO     46
CAL=IRALST(LA);                                                                             REO     47
L=LG(',',B,A);                                                                              REO     48
IF PRNTI THEN CALL STRPN(L,0);                                                              REO     49
BETA=L; CALL CANONCL(L,MOD);                                                                REO     50
BA,LA=L;                                                                                    REO     51
LC=LG('U',X,C);                                                                             REO     52
BETA=LC; CALL CANONCL(LC,MOD);                                                              REO     53
LA=LG(':',L,LC);                                                                            REO     54
IF PRNTI THEN CALL STRPN(LA,0);                                                             REO     55
BETA=LA; CALL CANONCL(LA,MOD);                                                              REO     56
N=(MOD-1-WTA)/WTB; KC=(9)'0' ||'1';                                                         REO     57
BABB=0;                                                                                     REO     58
DO J=2 TO N;                                                                                REO     59
   KC=KC+'1'; KT=SUBSTR(KC,8,2);                                                            REO     60
   IF INT THEN IF NX<KT THEN GOTO S1;                                                       REO     61
   L=LG(',',L,B);                                                                           REO     62
   IF PRNTI THEN CALL STRPN(L,0);                                                           REO     63
   BETA=L; CALL CANONCL(L,MOD);                                                             REO     64
   IF J=2 THEN BAB=L;  IF J=3 THEN BABB=L;                                                  REO     65
   IF INT THEN IF NX=KT THEN DO;                                                            REO     66
      LB=LIST(9); CAL=NXTLFT(UNSPEC('0I01'),LB);                                            REO     67
      LB=LG(':',L,LG('U',LB,0));                                                            REO     68
      IF PRNTI THEN CALL STRPN(LB,0);                                                       REO     69
      BETA=LB; CALL CANONCL(LB,MOD);                                                        REO     70
      LA=LG('.',LA,LB);                                                                     REO     71
      IF PRNTI THEN CALL STRPN(LA,0);                                                       REO     72
      GOTO S1;                    END;                                                      REO     73
   IF INT THEN LB=NCR(NX,KT);                                                               REO     74
      ELSE DO; LB=LIST(9); CAL=NXTLFT(UNSPEC('0I'||KT),LB);                                 REO     75
               LB=LG('|',X,LB);                                                             REO     76
```

```
                    END;                                                    REO     77
          LB=LG(':',L,LG('U',LB,0));                                        REO     78
          IF PRNTI THEN CALL STRPN(LB,0);                                   REO     79
          BETA=LB; CALL CANONCL(LB,MOD);                                    REO     80
          LA=LG('.',LA,LB);                                                 REO     81
        END;                                                                REO     82
                                                                            REO     83
S1:     IF 2*WTA+3*WTB < MOD THEN DO;                                       REO     84
          L=LG(',',BAB,BA);                                                 REO     85
          BETA=L; CALL CANONCL(L,MOD);                                      REO     86
          L1=LIST(9); CAL=NXTLFT(UNSPEC('0101'),L1);                        REO     87
          L3=LIST(9); CAL=NXTLFT(UNSPEC('0103'),L3);                        REO     88
          IF INT THEN DO;                                                   REO     89
            KT=SUBSTR(CHAR(NX+'01'),8,2);                                   REO     90
            LC=NCR(KT,'03');                                                REO     91
            BETA=LC; CALL CANONCL(LC,MOD);                                  REO     92
                    END;                                                    REO     93
                ELSE DO;                                                    REO     94
            LC=LG('|',LG('+',X,L1),L3);                                     REO     95
                    END;                                                    REO     96
          L=LG(':',L,LC);                                                   REO     97
          BETA=L; CALL CANONCL(L,MOD);                                      REO     98
          LA=LG('.',LA,L);                                                  REO     99
                                    END;                                    REO    100
        IF 2*WTA+4*WTB < MOD THEN DO;                                       REO    101
          IF BABB=0 THEN GOTO S2;                                           REO    102
          L=LG(',',BABB,BA);                                                REO    103
          BETA=L; CALL CANONCL(L,MOD);                                      REO    104
          L2=LIST(9); CAL=NXTLFT(UNSPEC('0102'),L2);                        REO    105
          L4=LIST(9); CAL=NXTLFT(UNSPEC('0104'),L4);                        REO    106
          IF INT THEN DO;                                                   REO    107
            LC=NCR(KT,'04');                                                REO    108
            BETA=LC; CALL CANONCL(LC,MOD);                                  REO    109
                    END;                                                    REO    110
```

```
            ELSE DO;                                                    REO   111
          LC=LG('}',LG('+',X,L1),L4);                                   REO   112
                END;                                                    REO   113
        L=LG(':',L,LC);                                                 REO   114
        BETA=L; CALL CANONCL(L,MOD);                                    REO   115
        LA=LG('.',LA,L);                                                REO   116
                              END;                                      REO   117
                                                                        REO   118
S2:     BETA=LA;                                                        REO   119
        CALL SCNP(LA,LA,0,MOD);                                         REO   120
        IF PRNTI THEN CALL STRPN(LA,0);                                 REO   121
        CAL= IRALST(INLSTR(LA,MTLIST(STR)));                            REO   122
        BETA=BETS;                                                      REO   123
        IF PRNTI THEN PUT SKIP(2) LIST('EXIT REO');                     REO   124
        END REO;                                                        REO   125


RINVOUT: PROC(STR,MOD);                                                 RINVOUT  0
        /* (A,B')-> (A,B)'(B,A,B')     */                              RINVOUT  1
        DCL (                                                           RINVOUT  2
             A,B,L,CAL,STR,MOD                                          RINVOUT  3
            )     FIXED BIN(31,0);                                      RINVOUT  4
        DCL (                                                           RINVOUT  5
             LG                                                         RINVOUT  6
            )ENTRY(CHAR(1),FIXED BIN(31,0),FIXED BIN(31,0))             RINVOUT  7
             RETURNS(FIXED BIN(31,0));                                  RINVOUT  8
        DCL (                                                           RINVOUT  9
             BOT,CON),WT,MTLIST                                         RINVOUT 10
            )ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));           RINVOUT 11
        DCL (                                                           RINVOUT 12
             MADNTP,INLSTR                                              RINVOUT 13
            )ENTRY(FIXED BIN(31,0),FIXED BIN(31,0))                     RINVOUT 14
             RETURNS(FIXED BIN(31,0));                                  RINVOUT 15
```

```
        A=CONT(MADNTP(STR,2)+1);                                          RINVOUT 16
        B=BOT(BOT(STR));                                                  RINVOUT 17
        L=LG('''',LG(',',A,B),0);                                         RINVOUT 18
        IF WT(A)+2*WT(B)<MOD THEN                                         RINVOUT 19
           L=LG('.',L,LG(',',LG(',',B,A),LG('''',B,0)));                  RINVOUT 20
        CALL IRALST(INLSTR(L,MTLIST(STR)));                               RINVOUT 21
        END RINVOUT;                                                      RINVOUT 22


RPRDOUT: PROC(STR,MOD);                                                   RPRDOUT  0
        /*  A,BC -> (A,C)(A,B)(A,B,C)  */                                 RPRDOUT  1
        DCL (                                                             RPRDOUT  2
             A,B,C,STR,L,CAL,MOD,WA,WB,WC,BETS,LA                         RPRDOUT  3
            )     FIXED BIN(31,0);                                        RPRDOUT  4
        DCL (                                                             RPRDOUT  5
             LG                                                           RPRDOUT  6
            )ENTRY(CHAR(1),FIXED BIN(31,0),FIXED BIN(31,0))               RPRDOUT  7
             RETURNS(FIXED BIN(31,0));                                    RPRDOUT  8
        DCL (                                                             RPRDOUT  9
             BOT,CONT,WT,LIST,MTLIST                                      RPRDOUT 10
            )ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));             RPRDOUT 11
        DCL (                                                             RPRDOUT 12
             MADNTP,NXTLFT,INLSTR                                         RPRDOUT 13
            )ENTRY(FIXED BIN(31,0),FIXED BIN(31,0))                       RPRDOUT 14
             RETURNS(FIXED BIN(31,0));                                    RPRDOUT 15
        DCL (STRPN,CANONCL) ENTRY (FIXED BIN(31,0),FIXED BIN(31,0));      RPRDOUT 16
        DCL PRNTI BIT(1) EXT,BETA FIXED BIN(31,0)EXT;                     RPRDOUT 17
        IF PRNTI THEN PUT SKIP(2) LIST('ENTER RPRDOUT');                  RPRDOUT 18
        IF PRNTI THEN CALL STRPN(STR,0);                                  RPRDOUT 19
        BETS=BETA;                                                        RPRDOUT 20
        A=CONT(MADNTP(STR,2)+1);                                          RPRDOUT 21
        B=CONT(MADNTP(BOT(STR),2)+1);                                     RPRDOUT 22
        C=BOT(BOT(STR));                                                  RPRDOUT 23
```

```
      WA=WT(A); WB=WT(B); WC=WT(C);                                    RPRDOUT 24
      IF WA+WB >= MOD THEN DO;                                         RPRDOUT 25
        L=LIST(9); CAL=NXTLFT(UNSPEC('0000'),L);  END;                 RPRDOUT 26
      IF WA+WB < MOD THEN DO;                                          RPRDOUT 27
         L=LG(',',A,B);                                                RPRDOUT 28
         IF PRNTI THEN CALL STRPN(L,0);                                RPRDOUT 29
         BETA=L; CALL CANONCL(L,MOD); END;                             RPRDOUT 30
      IF WA+WB+WC < MOD THEN DO;                                       RPRDOUT 31
         LA=LG(',',L,C);                                               RPRDOUT 32
         IF PRNTI THEN CALL STRPN(LA,0);                               RPRDOUT 33
         BETA=LA; CALL CANONCL(LA,MOD);                                RPRDOUT 34
         L=LG('.',L,LA);  END;                                         RPRDOUT 35
      IF WA+WC < MOD THEN DO;                                          RPRDOUT 36
         LA=LG(',',A,C);                                               RPRDOUT 37
         IF PRNTI THEN CALL STRPN(LA,0);                               RPRDOUT 38
         BETA=LA; CALL CANONCL(LA,MOD);                                RPRDOUT 39
         L=LG('.',LA,L);                                               RPRDOUT 40
           IF PRNTI THEN CALL STRPN(L,0);                              RPRDOUT 41
         BETA=L; CALL CANONCL(L,MOD);  END;                            RPRDOUT 42
      CALL IRALST(INLSTR(L,MTLIST(STR)));                              RPRDOUT 43
      IF PRNTI THEN PUT SKIP(2) LIST('EXIT RPRDOUT');                  RPRDOUT 44
      BETA=BETS;                                                       RPRDOUT 45
      END RPRDOUT;                                                     RPRDOUT 46


SCAN: PROC(STR,STROLD,POS,MOD)      RECURSIVE;                         SCAN     0
      /* POS=1 -> STR LEFT OF STROLD*/                                 SCAN     1
      /* POS=2 -> STR RGHT OF STROLD */                                SCAN     2
      /* POS=0 -> STR IS MAIN STRUCTURE */                             SCAN     3
      /* STR IS STRUCTURE TO BE SCANNED */                             SCAN     4
      /* STROLD IS STRUCTURE FROM WHICH STR IS DERIVED */              SCAN     5
      DCL (                                                            SCAN     6
           MOD, STR,STROLD,POS,L,R,CAL,MODL,MODR,N                     SCAN     7
```

```
     )     FIXED BIN(31,0);                                                    SCAN      8
DCL (                                                                          SCAN      9
     SYM                                                                       SCAN     10
     ) STR  CHAR(1);                                                           SCAN     11
DCL (                                                                          SCAN     12
     NTERM                                                                     SCAN     13
     )ENTRY(FIXED BIN(31,0)) RETURNS(BIT(1));                                  SCAN     14
DCL (                                                                          SCAN     15
     SYMB                                                                      SCAN     16
     )ENTRY(FIXED BIN(31,0)) RETURNS(CHAR(1));                                 SCAN     17
DCL (                                                                          SCAN     18
     BOT,CONT,WT                                                               SCAN     19
     )ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));                         SCAN     20
DCL(                                                                           SCAN     21
     CANONCL,STRPN                                                             SCAN     22
     )ENTRY(FIXED BIN(31,0),FIXED BIN(31,0));                                  SCAN     23
DCL (                                                                          SCAN     24
     MADNTP,SUBST,SUBSBT                                                       SCAN     25
     )ENTRY(FIXED BIN(31,0),FIXED BIN(31,0))                                   SCAN     26
     RETURNS(FIXED BIN(31,0));                                                 SCAN     27
DCL SCAN ENTRY(FIXED BIN(31,0),FIXED BIN(31,0),                                SCAN     28
              FIXED BIN(31,0),FIXED BIN(31,0));                                SCAN     29
IF¬ NTERM(STR) THEN RETURN;                                                    SCAN     30
MODL=MOD; MODR=MOD;                                                            SCAN     31
L=CONT(MADNTP(STR,2)+1); R=BOT(STR); SYM=SYMB(STR);                            SCAN     32
IF SYM='0' | SYM='U' THEN GOTO L1;                                             SCAN     33
IF SYM=',' THEN                                                                SCAN     34
   DO; MODL=MOD-WT(R); MODR=MOD-WT(L); END;                                    SCAN     35
IF NTERM(L) THEN CALL SCAN(L,STR,1,MODL);                                      SCAN     36
L1: IF NTERM(R) THEN CALL SCAN(R,STR,2,MODR);                                  SCAN     37
IF SYMB(STR)='.' & SYMB(STROLD)='.' & STR¬=STROLD THEN RETURN;                 SCAN     38
L2: CALL CANONCL(STR,MOD);                                                     SCAN     39
RETURN;                                                                        SCAN     40
END SCAN;                                                                      SCAN     41
```

```
STRGN:  PROC(STRNG)  FIXED BIN(31,0) RECURSIVE;                          STRGN    0
        /* AT THE FIRST LEVEL OF CALL THIS PROCEDURE TAKES A             STRGN    1
        STRING LIST STRNG, DIVIDES IT IN TWO PARTS SEPARATED BY          STRGN    2
        AN APPROPRIATE OPERATOR.  IT THEN GENERATES A STRUCTURE          STRGN    3
        BASED ON THESE TWO PARTS ANS THE DIVIDING OPERATOR.     */       STRGN    4
        DCL (                                                            STRGN    5
             F,S,SA,SB,SC,SD,BC,LR,PNT,STRNG,FA,CAL,P,N,PC,PNC           STRGN    6
            )     FIXED BIN(31,0);                                       STRGN    7
        DCL (                                                            STRGN    8
             NULSTR,SEQLR                                                STRGN    9
            )ENTRY(FIXED BIN(31,0),FIXED BIN(31,0))                      STRGN   10
             RETURNS(FIXED BIN(31,0));                                   STRGN   11
        DCL (                                                            STRGN   12
             CONT,                                                       STRGN   13
             SEQRDR,STRGN,LNKL,LNKR,POPTOP,POPBOT,DELETE,TOP,BOT         STRGN   14
            )ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));            STRGN   15
        DCL (                                                            STRGN   16
             CH,CHC,C,CC                                                 STRGN   17
            )     CHAR(1);                                               STRGN   18
        DCL (                                                            STRGN   19
             LG                                                          STRGN   20
            )ENTRY(CHAR(1),FIXED BIN(31,0),FIXED BIN(31,0))              STRGN   21
             RETURNS(FIXED BIN(31,0));                                   STRGN   22
        DCL PREC ENTRY(CHAR(1)) RETURNS(FIXED BIN(31,0));                STRGN   23
        DCL (                                                            STRGN   24
             NTERM                                                       STRGN   25
            )ENTRY(FIXED BIN(31,0)) RETURNS(BIT(1));                     STRGN   26
                                                                         STRGN   27
                                                                         STRGN   28
        IF ¬NTERM(STRNG) THEN RETURN(TOP(STRNG));                        STRGN   29
                                                                         STRGN   30
```

```
        IF SUBSTR(UNSPEC(BOT(STRNG)),25,8) = UNSPEC(')') &                  STRGN   31
           SUBSTR(UNSPEC(TOP(STRNG)),25,8) = UNSPEC('(') THEN               STRGN   32
           DO; CAL=POPTOP(STRNG); CAL=POPBOT(STRNG);                        STRGN   33
              RETURN(STRGN(STRNG));                                         STRGN   34
           END;                                                             STRGN   35
                                                                            STRGN   36
        BC=0; PC=10; CC='0';                                                STRGN   37
        S=SEQRDR(STRNG);                                                    STRGN   38
SCN:    PNT=LNKR(S); SA=SEQLR(S,FA);                                        STRGN   39
        IF FA>0 THEN DO;                                                    STRGN   40
           LR=NULSTR(PNC,STRNG); CAL=POPTOP(LR);                            STRGN   41
           /* ERASE READER */                                               STRGN   42
           RETURN(LG(CHC,STRGN(STRNG),STRGN(LR)));                          STRGN   43
                    END;                                                    STRGN   44
        IF FA=0 THEN GOTO SCN;                                              STRGN   45
        UNSPEC(CH)=SUBSTR(UNSPEC(SA),25,8);                                 STRGN   46
        IF CH='(' THEN DO; BC=BC+1; GOTO SCN; END;                          STRGN   47
        IF CH=')' THEN DO; BC=BC-1; GOTO SCN; END;                          STRGN   48
        IF BC<0 THEN SIGNAL CONDITION(ALPHA);                               STRGN   49
        IF BC>0 THEN GOTO SCN;                                              STRGN   50
        P=PREC(CH); UNSPEC(C)=SUBSTR(UNSPEC(SA),17,8);                      STRGN   51
        IF CH=',' & P=PC THEN                                               STRGN   52
           IF C>=CC THEN                                                    STRGN   53
                 DO; CC=C; PNC=PNT; CHC=CH; GOTO SCN; END;                  STRGN   54
                    ELSE GOTO SCN;                                          STRGN   55
        IF P<= PC THEN                                                      STRGN   56
           DO; PC=P; PNC=PNT; CHC=CH; CC=C; GOTO SCN; END;                  STRGN   57
        GOTO SCN;                                                           STRGN   58
PREC:   PROC(S) FIXED BIN(31,0);                                            STRGN   59
        DCL I FIXED BIN(31,0);                                              STRGN   60
        DCL S CHAR(1);                                                      STRGN   61
        DCL CH(9) CHAR(1) STATIC                                            STRGN   62
           INITIAL('U','-','+','|','/',',','.','*',':');                    STRGN   63
        DO I=1 TO 9; IF CH(I)=S THEN RETURN (I); END;                       STRGN   64
```

```
        PUT LIST('INVALID OPERATOR'); RETURN(20);                          STRGN  65
        END PREC;                                                          STRGN  66
         END STRGN;                                                        STRGN  67


STRNGET: PROC(LMAIN);                                                      STRNGET  0
        /* THIS ROUTINE TAKES COMMUTATOR EXPRESSION FROMCARD               STRNGET  1
        AND GENERATES THE CORRESPONDING LIST STRUCTURE.                    STRNGET  2
        BLANKS MAY BE USED FOR SPACING ARBITRARILY.                        STRNGET  3
        THE EXPRESSION MUST BE TERMINATED BY '#'.                          STRNGET  4
        THE LETTERS A -> L ARE ASSUMED TO BE GROUP ELEMENTS.               STRNGET  5
        THE LETTERS M -> Z ARE ASSUMED TO BE INTEGER LITERALS   */         STRNGET  6
        DCL (                                                              STRNGET  7
             GPL(12),INL(14),C,CH,CHA,CA                                   STRNGET  8
            )     CHAR(1);                                                 STRNGET  9
        DCL (                                                              STRNGET 10
             GPA(12),INA(14),L,LMAIN,I,J,CAL                               STRNGET 11
            )     FIXED BIN(31,0);                                         STRNGET 12
        DCL (                                                              STRNGET 13
             NXTLFT,SUBSBT                                                 STRNGET 14
            )ENTRY(FIXED BIN(31,0),FIXED BIN(31,0))                        STRNGET 15
             RETURNS(FIXED BIN(31,0));                                     STRNGET 16
        DCL (                                                              STRNGET 17
             LIST,STRGN,NAMTST,BOT                                         STRNGET 18
            )ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));              STRNGET 19
        DCL  CHT CHAR(10), CHF CHAR(4);                                    STRNGET 20
        GPL=' '; GPA=0; INL=' '; INA=0; K=0;                               STRNGET 21
        CAL=LIST(LMAIN); J=0; CH=' ';                                      STRNGET 22
  NUCH: IF J=1 THEN DO; J=0; GOTO SORT; END;                               STRNGET 23
  NXT:  GET EDIT(C)(A(1)); IF C=' ' THEN GOTO NXT;                         STRNGET 24
        IF C='#' THEN RETURN;                                              STRNGET 25
        IF CH>= 'A' & CH<= 'Z' THEN                                        STRNGET 26
           IF C='(' | (C>='A' & C<= 'Z') THEN                              STRNGET 27
```

```
            DO; J=1; CH='.'; GOTO OPRTR; END;                                  STRNGET 28
         IF CH=')' & ((C>='A' & C<='Z') | C='(') THEN                          STRNGET 29
            DO; J=1; CH='.'; GOTO OPRTR; END;                                  STRNGET 30
  SORT: CH=C;                                                                  STRNGET 31
         IF C>'Z' THEN GOTO INT;                                               STRNGET 32
         K=0;                                                                  STRNGET 33
         IF C<'A' THEN GOTO OPRTR;                                             STRNGET 34
         IF C<'M' THEN GOTO GPEL;                                              STRNGET 35
         IF C<'0' THEN GOTO INTLIT;                                            STRNGET 36
         SIGNAL CONDITION(ALPHA);                                              STRNGET 37
OPRTR:   I=BOT(LMAIN);                                                         STRNGET 38
         IF NAMTST(I)=0 THEN GOTO LA;                                          STRNGET 39
         UNSPEC(CHA)=SUBSTR(UNSPEC(I),25,8);                                   STRNGET 40
         IF CH=',' & CHA=',' THEN DO;                                          STRNGET 41
            UNSPEC(CHF)=UNSPEC(I);                                             STRNGET 42
            CHT='000000000' || SUBSTR(CHF,3,1); CHT=CHT+'1';                   STRNGET 43
            I=UNSPEC('00' || SUBSTR(CHT,9,1) || CHA);                          STRNGET 44
            CAL=SUBSBT(I,LMAIN); GOTO NUCH;                                    STRNGET 45
                  END;                                                         STRNGET 46
   LA:   CAL=NXTLFT(UNSPEC('000'||CH),LMAIN); GOTO NUCH;                       STRNGET 47
 GPEL:   DO I=1 TO 12;                                                         STRNGET 48
         IF GPL(I)=CH THEN                                                     STRNGET 49
            DO; CAL=NXTLFT(GPA(I),LMAIN); GOTO NUCH; END;                      STRNGET 50
         IF GPL(I)=' ' THEN DO;                                                STRNGET 51
            CAL=NXTLFT(UNSPEC('0GO'||CH),LIST(L));                             STRNGET 52
            GPL(I)=CH; CAL=NXTLFT(L,LMAIN); GPA(I)=L; GOTO NUCH; END;          STRNGET 53
                                 END;                                          STRNGET 54
         SIGNAL CONDITION(ALPHA);                                              STRNGET 55
INTLIT:  DO I=1 TO 14;                                                         STRNGET 56
         IF INL(I)=CH THEN                                                     STRNGET 57
            DO; CAL=NXTLFT(INA(I),LMAIN); GOTO NUCH; END;                      STRNGET 58
         IF INL(I)=' ' THEN DO;                                                STRNGET 59
            CAL=NXTLFT(UNSPEC('0LO'||CH),LIST(L));                             STRNGET 60
            INL(I)=CH; INA(I)=L;                                               STRNGET 61
```

```
          CAL=NXTLFT(L,LMAIN); GOTO NUCH; END;                                  STRNGET 62
                                  END;                                          STRNGET 63
        SIGNAL CONDITION(ALPHA);                                                STRNGET 64
INT:    IF K=1 THEN GOTO INB;                                                   STRNGET 65
        K=1; CAL=NXTLFT(UNSPEC('010' || CA),LMAIN); GOTO NUCH;                  STRNGET 66
INB:    K=0; CAL =NXTLFT(UNSPEC('01' || C || CA),LMAIN); GOTO NUCH;             STRNGET 67
        END STRNGET;                                                            STRNGET 68


STRPN:  PROC(STR,N)          RECURSIVE;                                         STRPN    0
        /* THIS PROC PRINTS A STRUCTURE, STR, SUPPLIED IN                      STRPN    1
           NAME FORMAT. LEFT NORMING OF COMMUTATORS IS ASSUMED. */              STRPN    2
        DCL (                                                                   STRPN    3
             A,B,CAL,I,NCOM,STR,N,PA,PB,PS                                      STRPN    4
            )     FIXED BIN(31,0);                                              STRPN    5
        DCL (                                                                   STRPN    6
             SYMB                                                               STRPN    7
            )ENTRY(FIXED BIN(31,0)) RETURNS(CHAR(1));                           STRPN    8
        DCL (                                                                   STRPN    9
             SYM                                                                STRPN   10
            )     CHAR(1);                                                      STRPN   11
        DCL (                                                                   STRPN   12
             COMCNT                                                             STRPN   13
            )ENTRY(FIXED BIN(31,0)) RETURNS(CHAR(1));                           STRPN   14
        DCL (                                                                   STRPN   15
             PREC                                                               STRPN   16
            )ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));                   STRPN   17
        DCL (                                                                   STRPN   18
             NTERM                                                              STRPN   19
            )ENTRY(FIXED BIN(31,0)) RETURNS(BIT(1));                            STRPN   20
        DCL (                                                                   STRPN   21
             BOT,CONT,TOP                                                       STRPN   22
            )ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));                   STRPN   23
```

```
 DCL (                                                                  STRPN   24
      MADNTP,STRPN                                                      STRPN   25
     )ENTRY(FIXED BIN(31,0),FIXED BIN(31,0))                            STRPN   26
      RETURNS(FIXED BIN(31,0));                                         STRPN   27
 DCL IDNT ENTRY(FIXED BIN(31,0)) RETURNS(BIT(1));                       STRPN   28
 IF N>500 THEN                                                          STRPN   29
    DO; PUT SKIP(N-500); CAL=COMCNT(STR); N=1; END;                     STRPN   30
 IF N=500 THEN DO; CAL=COMCNT(STR); N=1;END;                            STRPN   31
 IF N>200 THEN DO; N=N-200; CAL=COMCNT(STR); END;                       STRPN   32
 IF N>87 THEN DO; PUT SKIP(2); N=1; END;                                STRPN   33
IF N=0 THEN DO; PUT SKIP(3); CAL=COMCNT(STR); N=1;  END;                STRPN   34
SYM=SYMB(STR);                                                          STRPN   35
IF ¬NTERM(STR) THEN DO;                                                 STRPN   36
   IF IDNT(STR) THEN DO; PUT EDIT('1')(A); N=N+1;RETURN; END;           STRPN   37
 IF SUBSTR(UNSPEC(TOP(STR)),9,8)=UNSPEC('1') THEN DO;                   STRPN   38
    UNSPEC(SYM)=SUBSTR(UNSPEC(TOP(STR)),17,8);                          STRPN   39
 IF SYM¬='0' & SYM¬=' '  THEN DO; PUT EDIT(SYM)(A); N=N+1; END;         STRPN   40
    SYM=SYMB(STR); PUT EDIT(SYM)(A); N=N+1; RETURN; END;                STRPN   41
   PUT EDIT(SYM)(A); N=N+1; RETURN;                                     STRPN   42
                    END;                                                STRPN   43
A=CONT(MADNTP(STR,2)+1); B=BOT(STR);                                    STRPN   44
PS=PREC(STR); IF SYM=':' THEN PS=PS+1;                                  STRPN   45
PA=PREC(A); PB=PREC(B);                                                 STRPN   46
                                                                        STRPN   47
 IF SYM='''' | SYM='U' THEN GOTO L1;   /*UNARY OPERATION */            STRPN   48
                                                                        STRPN   49
/* STR IS NON TERMINAL & BINARY */                                      STRPN   50
IF PS>PA THEN DO; PUT EDIT('(')(A); N=N+1; END;                         STRPN   51
CAL=STRPN(A,N);                                                         STRPN   52
IF PS>PA THEN DO; PUT EDIT(')')(A); N=N+1; END;                         STRPN   53
 IF SYM='.' THEN GOTO L2;                                               STRPN   54
IF SYM¬=',' THEN DO;                                                    STRPN   55
   PUT EDIT(SYM)(A); N=N+1;  GOTO L2; END;                              STRPN   56
NCOM=(27)'0'B || SUBSTR(UNSPEC(TOP(STR)),21,4);                         STRPN   57
```

```
      DO I=1 TO NCOM; PUT EDIT(',')(A); N=N+1; END;                    STRPN   58
   L2: IF PS>PB THEN DO; PUT EDIT('(')(A); N=N+1; END;                 STRPN   59
      CAL=STRPN(B,N);                                                  STRPN   60
      IF PS>PB THEN DO; PUT EDIT(')')(A); N=N+1; END;                  STRPN   61
      RETURN;                                                          STRPN   62
   L1: IF SYM='''' THEN DO;                                            STRPN   63
      IF PS>PA THEN DO; PUT EDIT('(')(A); N=N+1; END;                  STRPN   64
      CAL=STRPN(A,N);                                                  STRPN   65
      IF PS>PA THEN DO; PUT EDIT(')')(A); N=N+1; END;                  STRPN   66
      PUT EDIT('''')(A); N=N+1; RETURN; END;                           STRPN   67
      IF SYM='U' THEN DO;                                              STRPN   68
      PUT EDIT('-')(A); N=N+1;                                         STRPN   69
      IF PS>PA THEN DO; PUT EDIT('(')(A); N=N+1; END;                  STRPN   70
      CAL=STRPN(A,N);                                                  STRPN   71
      IF PS>PA THEN DO; PUT EDIT(')')(A); N=N+1; END;                  STRPN   72
      RETURN;     END;                                                 STRPN   73
      END STRPN;                                                       STRPN   74


SUM:    PROC(A,B) FIXED BIN(31,0);                                     SUM      0
        /* ADDS TWO INTEGERS IN LIST FORMAT AND RETURNS THE SUM        SUM      1
         IN LIST FORMAT */                                             SUM      2
        DCL (AM,BM) BIT(1);                                            SUM      3
        DCL (DL,DR) FIXED DEC(5,0);                                    SUM      4
        DCL (A,B,L,R,CAL) FIXED BIN(31,0);                             SUM      5
        DCL CT CHAR(2);                                                SUM      6
        DCL(                                                           SUM      7
             LIST,BOT                                                  SUM      8
            )ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));          SUM      9
        DCL (                                                          SUM     10
             SYMB                                                      SUM     11
            )ENTRY(FIXED BIN(31,0)) RETURNS(CHAR(1));                  SUM     12
        DCL (                                                          SUM     13
```

```
     NTERM                                                              SUM     14
    )ENTRY(FIXED BIN(31,0)) RETURNS(BIT(1));                            SUM     15
DCL (                                                                   SUM     16
     NXTLFT                                                             SUM     17
    )ENTRY(FIXED BIN(31,0),FIXED BIN(31,0))                             SUM     18
     RETURNS(FIXED BIN(31,0));                                          SUM     19
DCL (                                                                   SUM     20
     LG                                                                 SUM     21
    )ENTRY(CHAR(1),FIXED BIN(31,0),FIXED BIN(31,0))                     SUM     22
     RETURNS(FIXED BIN(31,0));                                          SUM     23
AM='0'B; BM='0'B;                                                       SUM     24
IF NTERM(A) THEN IF SYMB(A)='U' THEN                                    SUM     25
                    DO; AM='1'B; L=BOT(BOT(A)); END;                    SUM     26
                                ELSE SIGNAL CONDITION(ALPHA);           SUM     27
            ELSE L=BOT(A);                                              SUM     28
IF NTERM(B) THEN IF SYMB(B)='U' THEN                                    SUM     29
                    DO; BM='1'B; R=BOT(BOT(B)); END;                    SUM     30
                                ELSE SIGNAL CONDITION(ALPHA);           SUM     31
            ELSE R=BOT(B);                                              SUM     32
UNSPEC(CT)=SUBSTR(UNSPEC(L),17,16);                                     SUM     33
DL=CT; IF AM THEN DL=-DL;                                               SUM     34
UNSPEC(CT)=SUBSTR(UNSPEC(R),17,16);                                     SUM     35
DR=CT; IF BM THEN DR=-DR;                                               SUM     36
DL=DL+DR;                                                               SUM     37
IF DL>99 THEN SIGNAL CONDITION(ALPHA);                                  SUM     38
IF DL=0 THEN DO;                                                        SUM     39
   R=LIST(9); CAL=NXTLFT(UNSPEC('0100'),R);                             SUM     40
  RETURN(R); END;                                                       SUM     41
IF DL<0 THEN DO;                                                        SUM     42
   DL=-DL;                                                              SUM     43
  R=LIST(9); CAL=NXTLFT(UNSPEC('01'||SUBSTR(CHAR(DL),7,2)),R);          SUM     44
   R=LG('U',R,0);                                                       SUM     45
   RETURN(R); END;                                                      SUM     46
IF DL>0 THEN DO;                                                        SUM     47
```

```
        R=LIST(9); CAL=NXTLFT(UNSPEC('01'||SUBSTR(CHAR(DL),7,2)),R);      SUM     48
        RETURN(R);                                                        SUM     49
      SIGNAL CONDITION (ALPHA);                                           SUM     50
      END SUM;                                                            SUM     51


SUMEX:  PROC(STR)  FIXED BIN(31,0);                                       SUMEX    0
      /* (A:M)(A:N) -> A:(M+N)  */                                        SUMEX    1
      /* UNARY MINUS IS REPLACED BY BINARYMINUS OR RAISED                 SUMEX    2
        TO THE TOP OF THE RESULTING INTEGRAL STRUCTURE     */             SUMEX    3
      DCL (                                                               SUMEX    4
           STR,A,B,M,N,L,R,CAL                                            SUMEX    5
          )     FIXED BIN(31,0);                                        SUMEX    6
      DCL (                                                               SUMEX    7
           BOT,TOP,CONT,MTLIST,LIST                                       SUMEX    8
          )ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));               SUMEX    9
      DCL (                                                               SUMEX   10
           LG                                                             SUMEX   11
          )ENTRY(CHAR(1),FIXED BIN(31,0),FIXED BIN(31,0))                 SUMEX   12
           RETURNS(FIXED BIN(31,0));                                      SUMEX   13
      DCL (                                                               SUMEX   14
           MADNTP,INLSTR,NXTLFT                                           SUMEX   15
          )ENTRY(FIXED BIN(31,0),FIXED BIN(31,0))                         SUMEX   16
           RETURNS(FIXED BIN(31,0));                                      SUMEX   17
      DCL (                                                               SUMEX   18
           LU,RU                                                          SUMEX   19
          )     BIT(1);                                                 SUMEX   20
      DCL IRALST ENTRY(FIXED BIN(31,0));                                  SUMEX   21
      DCL(                                                                SUMEX   22
           SYMB                                                           SUMEX   23
          )ENTRY(FIXED BIN(31,0)) RETURNS(CHAR(1));                       SUMEX   24
      DCL NTERM ENTRY(FIXED BIN(31,0)) RETURNS(BIT(1));                   SUMEX   25
      DCL (GL,GR) BIT(1);                                                 SUMEX   26
```

```
      GL,GR='0'B;                                                          SUMEX   27
      L=CONT(MADNTP(STR,2)+1);                                             SUMEX   28
      IF SYMB(L)=':' THEN A=CONT(MADNTP(L,2)+1);                           SUMEX   29
                     ELSE A=L;                                             SUMEX   30
      IF SYMB(A)='''' THEN DO;A=BOT(A); GL='1'B; END;                      SUMEX   31
      IF SYMB(L)¬=':'                                                      SUMEX   32
         THEN DO; M=LIST(9); CAL=NXTLFT(UNSPEC('0101'),M); END;            SUMEX   33
         ELSE M=BOT(L);                                                    SUMEX   34
      R=BOT(STR);                                                          SUMEX   35
      IF SYMB(R)¬=':'                                                      SUMEX   36
         THEN DO; N=LIST(9); CAL=NXTLFT(UNSPEC('0101'),N);                 SUMEX   37
                  IF SYMB(R)='''' THEN GR='1'B;                            SUMEX   38
              END;                                                         SUMEX   39
         ELSE DO; N=BOT(R);                                                SUMEX   40
                  IF SYMB(CONT(MADNTP(R,2)+1))='''' THEN GR='1'B;          SUMEX   41
              END;                                                         SUMEX   42
      LU='0'B;  RU='0'B;                                                   SUMEX   43
      IF SUBSTR(UNSPEC(TOP(M)),9,24)=UNSPEC('OOU') THEN LU='1'B;           SUMEX   44
      IF SUBSTR(UNSPEC(TOP(N)),9,24)=UNSPEC('OOU') THEN RU='1'B;           SUMEX   45
      IF GL & ¬LU THEN DO; M=LG('U',M,0); LU='1'B; GOTO K1; END;           SUMEX   46
      IF GL & LU THEN DO; M=BOT(M); LU='0'B; END;                          SUMEX   47
K1:   IF GR & ¬RU THEN DO; N=LG('U',N,0); RU='1'B; GOTO K2; END;           SUMEX   48
      IF GR & RU THEN DO; N=BOT(N); RU='0'B; END;                          SUMEX   49
K2:                                                                        SUMEX   50
      IF LU & RU THEN B=LG('U',LG('+',BOT(M),BOT(N)),0);                   SUMEX   51
      IF LU & ¬RU THEN B=LG('-',N,BOT(M));                                 SUMEX   52
      IF ¬LU & RU THEN B=LG('-',M,BOT(N));                                 SUMEX   53
      IF ¬LU & ¬RU THEN B=LG('+',M,N);                                     SUMEX   54
      CAL= IRALST(INLSTR(LG(':',A,B),MTLIST(STR)));                        SUMEX   55
      END SUMEX;                                                           SUMEX   56

SUMINT: PROC(STR);                                                         SUMINT   0
```

```
/* STR IS A STRUCTURE WITH OPERATOR '+' OR '-'. IT IS ASSUMED THAT          SUMINT   1
      THE LEFT AND RIGHT PARTS ARE IN CANONIC FORM AND IT                   SUMINT   2
      CONVERTS THE STRUCTURE TO CANONIC FORM.                    */         SUMINT   3
      DCL(                                                                  SUMINT   4
        L,R,BR,CAL,STR                                                      SUMINT   5
        )FIXED BIN(31,0);                                                   SUMINT   6
      DCL(                                                                  SUMINT   7
        TL,TR                                                               SUMINT   8
      )CHAR(4);                                                             SUMINT   9
      DCL(                                                                  SUMINT  10
        SUBST,MADNTP,INLSTR,SUM,SUBSTP                                      SUMINT  11
       )ENTRY(FIXED BIN(31,0),FIXED BIN(31,0))                              SUMINT  12
        RETURNS(FIXED BIN(31,0));                                           SUMINT  13
      DCL(                                                                  SUMINT  14
        LG                                                                  SUMINT  15
       )ENTRY(CHAR(1),FIXED BIN(31,0),FIXED BIN(31,0))                      SUMINT  16
        RETURNS(FIXED BIN(31,0));                                           SUMINT  17
      DCL(                                                                  SUMINT  18
        SYMB                                                                SUMINT  19
       )ENTRY(FIXED BIN(31,0)) RETURNS(CHAR(1));                            SUMINT  20
      DCL(                                                                  SUMINT  21
        CONT,MTLIST,TOP,BOT                                                 SUMINT  22
       )ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));                    SUMINT  23
      DCL (                                                                 SUMINT  24
      NTERM                                                                 SUMINT  25
      )ENTRY(FIXED BIN(31,0)) RETURNS(BIT(1));                              SUMINT  26
      DCL(WL,WR) CHAR(1);                                                   SUMINT  27
      DCL (LT,RT) CHAR(3);                                                  SUMINT  28
      DCL (WRA,WRB) CHAR(1);                                                SUMINT  29
      L=CONT(MADNTP(STR,2)+1); R=BOT(STR);                                  SUMINT  30
      IF SYMB(STR)='-' THEN DO;                                             SUMINT  31
         IF SYMB(R)='U' THEN DO;                                            SUMINT  32
            R=BOT(R); CAL=SUBSTP(UNSPEC('000+'),STR);                       SUMINT  33
                              END;                                          SUMINT  34
```

```
                        ELSE R=LG('U',R,0);                                    SUMINT  35
                          END;                                                 SUMINT  36
      UNSPEC(TL)=UNSPEC(TOP(L)); UNSPEC(TR)=UNSPEC(TOP(R));                    SUMINT  37
      WL=SUBSTR(TL,2,1); WR=SUBSTR(TR,2,1);                                    SUMINT  38
      LT=SUBSTR(TL,2,3); RT=SUBSTR(TR,2,3);                                    SUMINT  39
      IF WL='I' & WR='I' THEN                                                  SUMINT  40
         DO; L=SUM(L,R); GOTO RET; END;                                        SUMINT  41
      IF LT='OCU' & WR='I' THEN DO;                                            SUMINT  42
         UNSPEC(WRA)=SUBSTR(UNSPEC(TOP(BOT(L))),9,8);                          SUMINT  43
         IF WRA='I' THEN DO; L=SUM(L,R); GOTO RET; END;                        SUMINT  44
                          END;                                                 SUMINT  45
      IF WL='I' & RT='OOU' THEN DO;                                            SUMINT  46
         UNSPEC(WRB)=SUBSTR(UNSPEC(TOP(BOT(R))),9,8);                          SUMINT  47
         IF WRB='I' THEN DO; L=SUM(L,R); GOTO RET; END;                        SUMINT  48
                          END;                                                 SUMINT  49
      IF LT='OOU' & RT='OOU' THEN DO;                                          SUMINT  50
         UNSPEC(WRA)=SUBSTR(UNSPEC(TOP(BOT(L))),9,8);                          SUMINT  51
         UNSPEC(WRB)=SUBSTR(UNSPEC(TOP(BOT(R))),9,8);                          SUMINT  52
         IF WRA='I' &WRB='I' THEN DO;                                          SUMINT  53
           L=SUM(L,R); GOTO RET; END;                                          SUMINT  54
                          END;                                                 SUMINT  55
      IF ¬NTERM(L) & ¬NTERM(R) THEN RETURN;                                    SUMINT  56
      IF LT='100' | LT='I 0' THEN DO; L=R; GOTO RET; END;                      SUMINT  57
      IF RT='100' | RT='I 0' THEN GOTO RET;                                    SUMINT  58
      IF SUBSTR(TR,2,1)='I' THEN DO;  L=LG('+',R,L); GOTO RET; END;            SUMINT  59
      IF SUBSTR(TR,2,3)='OOU' THEN DO;                                         SUMINT  60
         BR=BOT(R); L=LG('-',L,BR);                                            SUMINT  61
         GOTO RET;                    END;                                     SUMINT  62
      RETURN;                                                                  SUMINT  63
RET:  CAL= IRALST(INLSTR(L,MTLIST(STR)));                                      SUMINT  64
      END SUMINT;                                                              SUMINT  65


SYMB: PROC(STR)          CHAR(1);                                              SYMB     0
```

```
        DCL (                                                          SYMB      1
          TOP                                                          SYMB      2
         )ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));             SYMB      3
        DCL (                                                          SYMB      4
          STR                                                          SYMB      5
         )      FIXED BIN(31,0);                                       SYMB      6
        DCL CH CHAR(1);                                                SYMB      7
        UNSPEC(CH)=SUBSTR(UNSPEC(TOP(STR)),25,8);                      SYMB      8
        RETURN(CH);                                                    SYMB      9
        END SYMB;                                                      SYMB     10


WT:     PROC(CMXP)          FIXED BIN(31,0)     RECURSIVE;             WT       0
        DCL (                                                          WT       1
          CMXP,WTT,WTB,WTM                                             WT       2
         )      FIXED BIN(31,0);                                       WT       3
        DCL (                                                          WT       4
          IDNT,GRPL                                                    WT       5
         )ENTRY(FIXED BIN(31,0)) RETURNS(BIT(1));                      WT       6
        DCL (                                                          WT       7
          SYMB                                                         WT       8
         )ENTRY(FIXED BIN(31,0)) RETURNS(CHAR(1));                     WT       9
        DCL (                                                          WT      10
          TOP,BOT,LOCT,CONT,WT                                         WT      11
         )ENTRY(FIXED BIN(31,0)) RETURNS(FIXED BIN(31,0));             WT      12
        DCL (                                                          WT      13
          MADNTP                                                       WT      14
         )ENTRY(FIXED BIN(31,0),FIXED BIN(31,0))                       WT      15
          RETURNS(FIXED BIN(31,0));                                    WT      16
        DCL (                                                          WT      17
          SYMBOL                                                       WT      18
         )      CHAR(1);                                              WT      19
        CMXP=LOCT(CMXP);                                               WT      20
```

```
 IF GRPL(CMXP)  THEN RETURN(1);           /* GROUP GENERATOR */          WT      21
 IF IDNT(CMXP) THEN RETURN('0'B||(31)'1'B); /*MAX POS NUMBER */          WT      22
 SYMBOL=SYMB(CMXP); WTM=WT(CONT(MADNTP(CMXP,2)+1));                      WT      23
IF SYMBOL='''' THEN RETURN (WTM);    /* INVERSE  */                      WT      24
IF SYMBOL=':' THEN RETURN(WTM);     /* EXP*/                            WT      25
WTB=WT(BOT(CMXP));                                                       WT      26
IF SYMBOL='*' THEN                                                       WT      27
   IF WTM < WTB THEN RETURN(WTM); ELSE RETURN(WTB); /*PROD*/             WT      28
IF SYMBOL=',' THEN RETURN(WTM+WTB);     /* COMMUTATOR  */                WT      29
 SIGNAL CONDITION(ALPHA);                                                WT      30
 END WT;                                                                 WT      31
```

```
(AB):4
ENTER ORDPROD
AB
AB
EXIT ORDPROD
(AB):4
ABABABAB
ENTER ORDPROD
ABABABAB
AAB(B,A)BABAB
AAAB(B,A)(B,A)(B,A,A)B(B,A)BAB
AAAAB(B,A)(B,A)(B,A,A)(B,A)(B,A,A)(B,A,A)(B,A,A,A)B(B,A)(B,A)(B,A,A)B(B,A)B
A:4B(B,A):2(B,A,A)(B,A)(B,A,A):2(B,A,A,A)B(B,A):2(B,A,A)B(B,A)B
ENTER LEO
(B,A):2,B
ENTER ORDPROD
(B,A,B):2(B,A,B,,B,A)
(B,A,B):2(B,A,B,,B,A)
```

```
EXIT ORDPROD
(B,A,B):2(B,A,B,,B,A)
EXIT LEO
ENTER LEO
(B,A,A):2,B
(B,A,A,B):2
EXIT LEO
A:488(B,A):2(B,A,B):2(B,A,B,,B,A)(B,A,A)(B,A,A,B)(B,A)(B,A,B)(B,A,A):2(B,A,A,B):2(B,A,A,
A)(B,A,A,A,B)(B,A):2(B,A,A)B(B,A)B
ENTER LEO
(B,A):2,B
ENTER ORDPROD
(E,A,B):2(B,A,B,,B,A)
(B,A,B):2(B,A,B,,B,A)
EXIT ORDPROD
(B,A,B):2(B,A,B,,B,A)
EXIT LEO
ENTER LEO
```

```
(B,A,B):2,B
(B,A,B,B):2
EXIT LEO
ENTER LEO
(B,A,A):2,B
(B,A,A,B):2
EXIT LEO
ENTER LEO
(B,A,A,B):2,B
(B,A,A,B,B):2
EXIT LEO
ENTER LEO
(B,A):2,B
ENTER ORDPROD
(B,A,B):2(B,A,B,,B,A)
(B,A,B):2(B,A,B,,B,A)
EXIT ORDPROD
(B,A,B):2(B,A,B,,B,A)
```

```
EXIT LEO
A:4BBB(B,A):2(B,A,B):2(B,A,B,,B,A)(B,A,B):2(B,A,B,B):2(B,A,B,,B,A)(B,A,A)(B,A,A,B)(B,A,
A,B)(B,A,A,B,B)(B,A)(B,A,B)(B,A,B)(B,A,B,B)(B,A,A):2(B,A,A,B):2(B,A,A,B):2(B,A,A,B,B):2(
B,A,A,A)(B,A,A,A,B)(B,A,A,A,B)(B,A):2(B,A,B):2(B,A,B,,B,A)(B,A,A)(B,A,A,B)(B,A)B

ENTER LEO
(B,A):2,B
ENTER ORDPROD
(B,A,B):2(B,A,B,,B,A)
(B,A,B):2(B,A,B,,B,A)
EXIT ORDPROD
(B,A,B):2(B,A,B,,B,A)
EXIT LEO
(B,A,B):2,B
(B,A,B,B):2
EXIT LEO
ENTER LEO
(B,A,B):2,B
(B,A,B,B):2
```

```
EXIT LEO
ENTER LEO
(B,A,B,B):2,B
(B,A,B,B,B):2
EXIT LEO
ENTER LEO
(B,A,A):2,B
(B,A,A,B):2
EXIT LEO
ENTER LEO
(B,A,A,B):2,B
(B,A,A,B,B):2
EXIT LEO
ENTER LEO
(B,A,A,B):2,B
(B,A,A,B,B):2
EXIT LEO
ENTER LEO
```

(B,A):2,B

ENTER ORDPROD

(B,A,B):2(B,A,B,,B,A)

(B,A,B):2(B,A,B,,B,A)

EXIT ORDPROD

(B,A,B):2(B,A,B,,B,A)

EXIT LEO

ENTER LEO

(B,A,B):2,B

(B,A,B,B):2

EXIT LEO

A:4B:4(B,A):2(B,A,B):2(B,A,B,,B,A)(B,A,B):2(B,A,B,B):2(B,A,B,,B,A)(B,A,B):2(B,A,B,B):4(B,A,B,B,B):2(B,A,B,,B,A)(B,A,A)(B,A,A,B):2(B,A,A,B,B)(B,A,A,B)(B,A,A,B,B):2(B,A)(B,A,B):2(B,A,B,B)(B,A,B)(B,A,B,B):2(B,A,B,B,B)(B,A,A):2(B,A,A,B):4(B,A,A,B,B):2(B,A,A,B):2(B,A,A,B,B):4(B,A,A,A)(B,A,A,A,B):3(B,A):2(B,A,B):2(B,A,B,,B,A)(B,A,B):2(B,A,B,B):2(B,A,B,,B,A)(B,A,A)(B,A,A,B):2(B,A,A,B,B)(B,A)(B,A,B)

ENTER LEO

(B,A,B):2,,B,A

(B,A,B,,B,A):2

EXIT L EO

ENTER L EO

(B,A,B):2,,B,A

(B,A,B,,B,A):2

EXIT L EO

ENTER L EO

(B,A,B):2,,B,A

(B,A,B,,B,A):2

EXIT L EO

A:4B:4(B,A):2(B,A)(B,A,B):2(B,A,B,,B,A):2(B,A,B,,B,A)(B,A,B):2(B,A,B,,B,A):2(B,A,B,B):2(B,A,B,,B,A)(B,A,B):2(B,A,B,,B,A):2(B,A,B,B):4(B,A,B,B,B):2(B,A,B,,B,A)(B,A,A)(B,A,A,,B,A)(B,A,A,B):2(B,A,A,B,B)(B,A,A,B)(B,A,A,B,B):2(B,A,B):2(B,A,B,B)(B,A,B)(B,A,B,B):2(B,A,B,B,B)(B,A,A):2(B,A,A,B):4(B,A,A,B,B):2(B,A,A,B):2(B,A,A,B,B):4(B,A,A,A)(B,A,A,A,B):3(B,A):2(B,A,B):2(B,A,B,,B,A)(B,A,B):2(B,A,B,B):2(B,A,B,,B,A)(B,A,A)(B,A,A,B):2(B,A,A,B,B)(B,A)(B,A,B)

ENTER R EO

(B,A,B):2,(B,A):2

ENTER R EO

B,A,(B,A,B):2

```
(B,A,B,,B,A):-2
EXIT REO
(B,A,B,,B,A):4
EXIT REO
ENTER REO
(B,A,B):2,(B,A):2
ENTER REO
B,A,(B,A,B):2
(B,A,B,,B,A):-2
EXIT REO
(B,A,B,,B,A):4
EXIT REO
ENTER REO
(B,A,B):2,(B,A):2
ENTER REO
B,A,(B,A,B):2
(B,A,B,,B,A):-2
EXIT REO
```

```
(B,A,B,,B,A):4
EXIT REO
ENTER REO
B,A,A,(B,A):2
(B,A,A,,B,A):2
EXIT REO
ENTER REO
(B,A,B):2,(B,A):2
ENTER REO
B,A,(B,A,B):2
(B,A,B,,B,A):-2
EXIT REO
(B,A,B,,B,A):4
EXIT REO
ENTER REO
B,A,B,(B,A):2
(B,A,B,,B,A):2
EXIT REO
```

```
ENTER REO
(B,A,A):2,(B,A):2
ENTER REO
B,A,(B,A,A):2
(B,A,A,,B,A):-2
EXIT REO
(B,A,A,,B,A):4
EXIT REO
A:4B:4(B,A):2(B,A)(B,A):2(B,A,B):2(B,A,B,,B,A):4(B,A,B,,B,A):2(B,A,B,,B,A)(B,A,B):2(B,A,
B,,B,A):4(B,A,B,,B,A):2(B,A,B,B):2(B,A,B,,B,A)(B,A,B):2(B,A,B,,B,A):4(B,A,B,,B,A):2(B,A,
B,B):4(B,A,B,B,B):2(B,A,B,,B,A)(B,A,A)(B,A,A,,B,A):2(B,A,A,,B,A)(B,A,A,B):2(B,A,A,B,B)(
B,A,A,B)(B,A,A,B,B):2(B,A,B):2(B,A,B,,B,A):4(B,A,B,B)(B,A,B)(B,A,B,,B,A):2(B,A,B,B):2(B,
A,B,B,B)(B,A,A):2(B,A,A,,B,A):4(B,A,A,B):4(B,A,A,B,B):2(B,A,A,B):2(B,A,A,B,B):4(B,A,A,A)
(B,A,A,A,B):3(B,A,B):2(B,A,B,,B,A)(B,A,B):2(B,A,B,B):2(B,A,B,,B,A)(B,A,A)(B,A,A,B):2(B,
A,A,B,B)(B,A)(B,A,B)

ENTER LEO
(B,A,B):2,,B,A
(B,A,B,,B,A):2
EXIT LEO
```

```
ENTER LEO
(B,A,B):2,,B,A
(B,A,B,,B,A):2
EXIT LEO
ENTER LEO
(B,A,B):2,,B,A
(B,A,B,,B,A):2
EXIT LEO
ENTER LEO
(B,A,B):2,,B,A
(B,A,B,,B,A):2
EXIT LEO
ENTER LEO
(B,A,A):2,,B,A
(B,A,A,,B,A):2
EXIT LEO
ENTER LEO
(B,A,B):2,,B,A
```

(B,A,B,,B,A):2

EXIT LEO

ENTER LEO

(B,A,B):2,,B,A

(B,A,B,,B,A):2

EXIT LEO

A:4B:4(B,A):2(B,A)(B,A):2(B,A)(B,A,B):2(B,A,B,,B,A):2(B,A,B,,B,A):4(B,A,B,,B,A):2(B,A,B,,B,A)(B,A,B):2(B,A,B,,B,A):2(B,A,B,,B,A):4(B,A,B,,B,A):2(B,A,B,B):2(B,A,B,,B,A)(B,A,B):2(B,A,B,,B,A):2(B,A,B,,B,A):4(B,A,B,,B,A):2(B,A,B,B):4(B,A,B,B,B):2(B,A,B,,B,A)(B,A,A)(B,A,A,,B,A)(B,A,A,,B,A):2(B,A,A,,B,A)(B,A,A,B):2(B,A,A,B,B)(B,A,A,B)(B,A,A,B,B):2(B,A,B):2(B,A,B,,B,A):2(B,A,B,,B,A):4(B,A,B,B)(B,A,B)(B,A,B,,B,A)(B,A,B,,B,A):2(B,A,B,B):2(B,A,B,B,B)(B,A,A):2(B,A,A,,B,A):2(B,A,A,,B,A):4(B,A,A,B):4(B,A,A,B,B):2(B,A,A,B):2(B,A,A,B,B):4(B,A,A,A)(B,A,A,A,B):3(B,A,B):2(B,A,B,,B,A):2(B,A,B,,B,A)(B,A,B):2(B,A,B,,B,A):2(B,A,B,B):2(B,A,B,,B,A)(B,A,A)(B,A,A,,B,A)(B,A,A,B):2(B,A,A,B,B)(B,A,B)

A:4B:4(B,A):6(B,A,B):2(B,A,B,,B,A):9(B,A,B):2(B,A,B,,B,A):8(B,A,B,B):2(B,A,B,,B,A)(B,A,B):2(B,A,B,,B,A):8(B,A,B,B):4(B,A,B,B,B):2(B,A,B,,B,A)(B,A,A)(B,A,A,,B,A):4(B,A,A,B):2(B,A,A,B,B)(B,A,A,B)(B,A,A,B,B):2(B,A,B):2(B,A,B,,B,A):6(B,A,B,B)(B,A,B)(B,A,B,,B,A):3(B,A,B,B):2(B,A,B,B,B)(B,A,A):2(B,A,A,,B,A):6(B,A,A,B):4(B,A,A,B,B):2(B,A,A,B):2(B,A,A,B,B):4(B,A,A,A)(B,A,A,A,B):3(B,A,B):2(B,A,B,,B,A):3(B,A,B):2(B,A,B,,B,A):2(B,A,B,B):2(B,A,B,,B,A)(B,A,A)(B,A,A,,B,A)(B,A,A,B):2(B,A,A,B,B)(B,A,B)

A:4B:4(B,A):6(B,A,A):4(B,A,B):2(B,A,B,,B,A):9(B,A,B):2(B,A,B,,B,A):8(B,A,B,B):2(B,A,B,,B,A)(B,A,B):2(B,A,B,,B,A):8(B,A,B,B):4(B,A,B,B,B):2(B,A,B,,B,A)(B,A,A,,B,A):4(B,A,A,B):2(B,A,A,B,B)(B,A,A,B)(B,A,A,B,B):2(B,A,B):2(B,A,B,,B,A):6(B,A,B,B)(B,A,B)(B,A,B,,B,A):3(B,A,B,B):2(B,A,B,B,B)(B,A,A,,B,A):6(B,A,A,B):4(B,A,A,B,B):2(B,A,A,B):2(B,A,A,B,B):4(B,A,A,A)(B,A,A,A,B):3(B,A,B):2(B,A,B,,B,A):3(B,A,B):2(B,A,B,,B,A):2(B,A,B,B):2(B,A,B,,B,A)(B,A,A,,B,A)(B,A,A,B):2(B,A,A,B,B)(B,A)

A:4B:4(B,A):6(B,A,A):4(B,A,B):14(B,A,B,,B,A):17(B,A,B,B):2(B,A,B,,B,A):9(B,A,B,B):4(B,A,B,B,B):2(B,A,B,,B,A)(B,A,A,,B,A):4(B,A,A,B):2(B,A,A,B,B)(B,A,A,B)(B,A,A,B,B):2(B,A,B,,B,A):6(B,A,B,B)(B,A,B,,B,A):3(B,A,B,B):2(B,A,B,B,B)(B,A,A,,B,A):6(B,A,A,B):4(B,A,A,B,B):2(B,A,A,B):2(B,A,A,B,B):4(B,A,A,A)(B,A,A,A,B):3(B,A,B,,B,A):5(B,A,B,B):2(B,A,B,,B,A)(B,A,A,,B,A)(B,A,A,B):2(B,A,A,B,B)

A:4B:4(B,A):6(B,A,A):4(B,A,B):14(B,A,A,A)(B,A,A,B):11(B,A,B,B):11(B,A,B,,B,A):26(B,A,B,B,B):2(B,A,B,,B,A)(B,A,A,,B,A):4(B,A,A,B,B):3(B,A,B,,B,A):9(B,A,B,B,B)(B,A,A,,B,A):6(B,A,A,B,B):6(B,A,A,A,B):3(B,A,B,,B,A):6(B,A,A,,B,A)(B,A,A,B,B)

A:4B:4(B,A):6(B,A,A):4(B,A,B):14(B,A,A,A)(B,A,A,B):11(B,A,B,B):11(B,A,A,,B,A):11(B,A,B,,B,A):26(B,A,B,B,B):2(B,A,B,,B,A)(B,A,A,B,B):3(B,A,B,,B,A):9(B,A,B,B,B)(B,A,A,B,B):6(B,A,A,A,B):3(B,A,B,,B,A):6(B,A,A,B,B)

A:4B:4(B,A):6(B,A,A):4(B,A,B):14(B,A,A,A)(B,A,A,B):11(B,A,B,B):11(B,A,A,,B,A):11(B,A,B,,B,A):42(B,A,B,B,B):2(B,A,A,B,B):3(B,A,B,B,B)(B,A,A,B,B):6(B,A,A,A,B):3(B,A,A,B,B)

```
A:4B:4(B,A):6(B,A,A):4(B,A,B):14(B,A,A,A)(B,A,A,B):11(B,A,B,B):11(B,A,A,,B,A):11(B,A,B,,
B,A):42(B,A,A,A,B):3(B,A,B,B,B):2(B,A,A,B,B):3(B,A,B,B,B)(B,A,A,B,B):7

A:4B:4(B,A):6(B,A,A):4(B,A,B):14(B,A,A,A)(B,A,A,B):11(B,A,B,B):11(B,A,A,,B,A):11(B,A,B,,
B,A):42(B,A,A,A,B):3(B,A,A,B,B):10(B,A,B,B,B):3

A:4B:4(B,A):6(B,A,A):4(B,A,B):14(B,A,A,A)(B,A,A,B):11(B,A,B,B):11(B,A,A,,B,A):11(B,A,B,,
B,A):42(B,A,A,A,B):3(B,A,A,B,B):10(B,A,B,B,B):3

EXIT ORCPROD

A:4B:4(B,A):6(B,A,A):4(B,A,B):14(B,A,A,A)(B,A,A,B):11(B,A,B,B):11(B,A,A,,B,A):11(B,A,B,,
B,A):42(B,A,A,A,B):3(B,A,A,B,B):10(B,A,B,B,B):3
```

15. REFERENCES.

Included in this list are several useful reports and papers which have not been referenced in the preceeding text.

BENDB650 Bender, B. MANIP - a computer system for algebra and analytic differentiation. N.B.S. Report 9115, Boulder laboratories, Colorado, (Nov., 1965).

BOBRD660 Bobrow, D.G., and Teitelman, W. Format-directed list processing in LISP. Report of Bolt, Beranek and Newman Inc., (April, 1966).

BRAFP630 Braffort, P., and Hirschberg, D. (Ed.) Computer Programming and Formal Systems. Nth. Holland Publishing Co. (1963) 33-70.

BROWW660 Brown, W.S. The ALTRAN language and the ALPAK system for symbolic algebra on a digital computer. Princeton University Summer Conference in Computer Sciences. (Aug. - Sept. 1966).

BROWW680 Brown, W.S. The ALTRAN language for symbolic algebra. Report of Bell Telephone Laboratories. (April, 1968).

CAMPJ660 Campbell, J.M., and Lamberth, W.J. Symbolic and numeric computation in group theory. Proc. Aust. Computer Conf., (1966).

CARRA660 Caracciolo di Forino, A., Spanedda, L., and Wolkenstein, N. PANON-1B: a programming language for symbol manipulation. SIC-SAM Symposium (March, 1966). Also Calcolo, 3 (April - June, 1966) 245-265.

GINSS661 Ginsburg, S. A sruvey of ALGOL-like and context-free language theory in Formal Language Description Languages for Computer Programming. Nth. Holland Publishing Co. (1966) 86-99.

HALLP340 Hall, P. A contribution to the theory of groups of prime-power order. Proc. Lond. Math. Soc. 36 (1934) 29-95.

HERMH650 Hermes, H. Enumerability, Decidability, Computability. Springer-Verlag (1965).

KRAUE640 Krause, E.F. On the collection process. Proc. American Math. Soc., 15 (June, 1964).

MAGNW660 Magnus, W.M., Karrass, A., and Solitar, D. Combinatorial Group Theory. John Wiley and Sons, (1966).

MARKA620 Markov, A.A., Theory of Algorithms. Israel Program for Scientific Translations, Jerusalem, (1962).

MCCAJ630 McCarthy, J. A basis for a mathematical theory of computation in Computer Programming and Formal Systems, Nth. Holland Publishing Co. (1963) 33-70.

MCCAJ650 McCarthy, J., et al. LISP 1.5 Programmers' Manual. Computation Lab. Report, M.I.T. (1965).

NAURP600 Naur, P. (Ed.) Report on the algorithmic language ALGOL-60. Comm. ACM, 3 (1960) 299-314.

PERLA660 Perlis, A.J., Iturriaga, R., and Standish, T.A. A definition of Formula ALGOL. Carnegie Institute of Technology, Pittsburgh, Pa., (March, 1966).

SAMMJ650 Sammet, J.E. An overall view of FORMAC. I.B.M., Systems Development Division, Poughkeepsie Laboratory Technical Report No. TROO. 1367, (Dec., 1965).

SAMMJ660 Sammet, J.E. Survey of use of computers for doing non-numerical mathematics. I.B.M., Systems Development Division, Poughkeepsie Laboratory Technical Report No. TROO. 1428. (March, 1966).

SAMMJ661 Sammet, J.E. An annotated descriptor based bibliography on the use of computers for non-numerical mathematics. Computing Reviews, 7 (July - Aug., 1966) B1-B31.

STEET660 Steel, T.B. Jr. (Ed.) Formal Language Description Languages for Computer Programming. Nth. Holland Publishing Co. (1966).

WARDM650 Ward, M.A. Basic Commutators for Polynilpotent Groups. Thesis. Australian National University.

WARDM Ward, M.A. Basic Commutators. To be published in Philos. Trans. Roy. Soc. London Ser.A.

WEIZJ630 Weizenbaum, J. Symmetric list processor. Comm. ACM 6 (Sept., 1963) 524-544.